



*Simplify and
Degunk
your Linux
configuration!*

*Clean up
your Linux
environment!*

Covers common
Linux distributions

DEgUNKing LiNux

Roderick W. Smith



PARAGLYPH
PRESS



Quick Degunking Sheet!

The Degunking 12-Step Program

Here is the basic 12-step degunking process that you'll follow in this book:

1. Delete files you don't need and better organize your remaining files (Chapter 3).
2. Select the desktop environment that best fits your needs and configure it properly to optimize the environment (Chapter 4).
3. Configure the settings for the common applications you use (Chapter 5).
4. Get rid of the unused packages on your system (Chapter 6).
5. Fine-tune the performance of the applications you use (Chapter 7).
6. Manage processes on your system, including dealing with misbehaving programs (Chapter 8).
7. Clean up the user accounts and system administration accounts on your system (Chapter 9).
8. Set up a system to reduce your e-mail spam and worms (Chapter 10).
9. Improve your network security (Chapter 10).
10. Fix any problems with hardware and drivers that you might have (Chapter 11).
11. Optimize your X configuration (Chapter 12).
12. Back up your system on a regular basis (Appendix A).

Degunking Linux with Time Limitations

If you're like most people, the words "I've got too much time on my hands" aren't likely to escape your lips. With this in mind, I'll present some important degunking tasks you can undertake, ordered by the time they're likely to consume. Of course, you should read through this entire book, or at least read the chapters that are most relevant to you; but if you want to degunk your system in chunks of limited time, this classification can help.

Fifteen-Minute Degunking

Some degunking tasks are very simple and straightforward. These tasks can usually be completed in fifteen minutes or less:

1. Learn to use your desktop environment's pager (page 97).
2. Select a new font for your Web browser (page 111).
3. Disable Java, and perhaps JavaScript, in your Web browser (page 114).
4. Locate and kill misbehaving programs (page 193).
5. Set a new password for yourself and for your root account (page 226).
6. Initiate a system backup (Appendix A, "Performing a Backup"). (Note: This activity assumes that your system is already configured to do backups.)

Thirty-Minute Degunking

If you've got half an hour to spare, you can perform several of the fifteen-minute degunking tasks or step it up a bit and tackle a job that will most likely take about thirty minutes. These tasks are more involved than the fifteen-minute tasks:

1. Add or delete OpenOffice.org fonts (page 104).
2. Reconfigure your OpenOffice.org printers, if they aren't working to your satisfaction (page 107).
3. Configure your Web browser's cookie settings and discard unwanted cookies (page 116).
4. Configure your e-mail client to display the features you want to use (page 121).
5. Check your system for signs of intrusion (page 259).
6. Set and test a new X color depth, resolution, or refresh rate (page 295).

One-Hour Degunking

One-hour degunking tasks are, naturally, more involved than the half-hour variety. They may involve some modest problem-solving or decision-making on your part, so don't undertake these tasks when you're drowsy or distracted:

1. Clean up your dot files (page 62).
2. Create an organized structure for your e-mail and sort your e-mail into appropriate folders (page 123).
3. Update your packages to the latest available versions (page 146).
4. Fix a misbehaving printer (page 171).
5. Identify and delete unnecessary accounts (page 219).
6. Install a degunking Web proxy and configure your Web browser to use it (page 238).
7. Track down and remove unnecessary servers (page 252).
8. Add or delete X fonts (page 302).

Three-Hour Degunking

Three-hour degunking tasks involve more serious problem solving or research to complete. You may need Internet access to perform these tasks, in order to download software or look up additional information on the Web:

1. Search for and delete unused files (page 48).
2. Tweak your GNOME or KDE configuration to optimize performance (page 70).
3. Investigate alternative office suites (page 109).
4. Investigate alternative Web browsers (page 119).
5. Investigate alternative e-mail clients (page 126).
6. Degunk a crashing or otherwise misbehaving program (page 164).
7. Configure a spam and worm filter for your e-mail (page 244).
8. Configure and compile a custom Linux kernel (page 278).

Half-Day Degunking

Beyond the three-hour mark are the half-day degunking tasks. These jobs require really rolling up your sleeves and cleaning the system out.

1. Try using a slimmer desktop environment, such as XFce (page 86).
2. Try using a bare window manager (page 89).
3. Use a package browser to identify and remove gunky packages on your system (page 133).
4. Buy, install, and configure a NAT router (page 257).
5. Replace unsupported hardware (page 286).
6. Configure the system to perform a backup (Appendix A).

Free Time Degunking

There may be times when you are doing something with your computer and you discover that you have some free time to spare. These tasks do not need to be completed in any specific order. Simply select a task and perform it to help improve the performance of your Linux system:

1. Locate and remove programs that are CPU or memory hogs (page 194).
2. Configure a spam and worm filter for your e-mail (page 244).
3. Check your system for signs of intrusion (page 259).
4. Search for and delete unused files (page 48).
5. Backup critical files that you haven't backed up in a while (page 316).
6. Create an organized structure for your e-mail and sort your e-mail into appropriate folders (page 123).

7. Delete user accounts that are no longer needed (page 219).
8. Organize your files by creating subdirectories (page 59).
9. Adjust the priorities of the processes running on your system to maximize CPU usage (page 202).
10. Investigate alternative Web browsers (page 119).
11. Track down and remove unnecessary servers (page 252).
12. Track your overall disk usage and fine-tune how you are using your directories (page 42).
13. Improve the performance of your printers (page 168).
14. Set your e-mail HTML and security options to better protect yourself from viruses (page 123).
15. Keep your GNOME or KDE desktop uncluttered (page 78 or page 84).
16. Clean up your e-mail by getting rid of messages that are no longer needed (page 244).
17. Try using a slimmer desktop environment, such as XFce (page 86).
18. Improve your system security by selecting good passwords (page 226).
19. Update your packages to the latest available versions (page 146).
20. Configure X fonts so that they display better on your screen (page 316).

DEgUNKING™ LiNUX

Roderick W. Smith

 **PARAGLYPH™**
PRESS

President **Degunking™ Linux**

*Keith
Weiskamp*

Copyright © 2005 Paraglyph Press. All rights reserved.

**Editor-at-
Large**

*Jeff
Duntemann*

This book may not be duplicated in any way without the express written consent of the publisher, except in the form of brief excerpts or quotations for the purposes of review. The information contained herein is for the personal use of the reader and may not be incorporated in any commercial programs, other books, databases, or any kind of software without written consent of the publisher. Making copies of this book or any portion for any purpose other than your own is a violation of United States copyright laws.

**Vice President,
Sales,
Marketing, and
Distribution**

Steve Sayre

Limits of Liability and Disclaimer of Warranty

The author and publisher of this book have used their best efforts in preparing the book and the programs contained in it. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book.

**Vice President,
International
Sales and
Marketing**

*Cynthia
Caldwell*

The author and publisher shall not be liable in the event of incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of the programs, associated instructions, and/or claims of productivity gains.

**Production
Manager**

Kim Eoff

Trademarks

Trademarked names appear throughout this book. Rather than list the names and entities that own the trademarks or insert a trademark symbol with each mention of the trademarked name, the publisher states that it is using the names for editorial purposes only and to the benefit of the trademark owner, with no intention of infringing upon that trademark.

**Cover
Designer**

Kris Sotelo

Paraglyph Press, Inc.

4015 N. 78th Street, #115

Scottsdale, Arizona 85251

Phone: 602-749-8787

www.paraglyphpress.com

Paraglyph Press ISBN: 1-933097-04-3

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

*This book is dedicated to the memory of my mother,
who was an expert at degunking around the house,
if not around the computer.*



Recently Published by Paraglyph Press:

Degunking Your Personal Finances

By Shannon Plate

Degunking Microsoft Office

*By Wayne Palaia
and Christina Palaia*

Degunking Your PC

*By Joli Ballew
and Jeff Dunteman*

Degunking eBay

By Greg Holden

Degunking Your Email, Spam, and Viruses

By Jeff Duntemann

Degunking Windows

*By Joli Ballew
and Jeff Duntemann*

Degunking Your Mac

By Joli Ballew

Acknowledgments

Although this book has one official author, it's not the work of just one person. Many people have contributed to this book in one way or another. Editor Keith Weiskamp has provided invaluable assistance on the book's tone. The copyeditor, Judy Flynn, has helped keep my prose readable. The layout and production team, headed by Kim Eoff, made the book visually appealing. Finally, my agent at Studio B, Neil Salkind, helped connect me with all of these people, making this book a possibility.

About the Author

Roderick W. Smith is a writer on computer technology. He has over a dozen books to his name, mostly about Linux and other open source software. These titles include *Advanced Linux Networking* (Addison-Wesley, 2002), *Linux PowerTools* (Sybex, 2003), and *Linux in a Windows World* (O'Reilly, 2005). He is also the author of *Linux Magazine's* monthly "Guru Guidance" column. His Web page is **<http://www.rodsbooks.com>**. Rod resides in Woonsocket, Rhode Island.

Contents

Introduction.....	xvii
--------------------------	-------------

Chapter 1

Why Is There Gunk in Linux?	1
--	----------

Learn How to Recognize Linux Gunk	2
<i>What Is Gunk?</i>	3
Understand the Common Types of Linux Gunk	5
<i>Gunky User Configurations</i>	5
<i>Gunky System Configurations</i>	8
<i>Gunky Hardware and Drivers</i>	10
Understand How Gunk Gets in Your System	11
<i>Out-of-the-Box Gunk</i>	11
<i>Gunk Added by Mistake</i>	14
<i>Gunk Added Maliciously</i>	14
Learn How to Eliminate Linux Gunk	15
Summing Up: A Taxonomy of Gunk	16

Chapter 2

Linux Degunking Strategies.....	17
--	-----------

The Strategy for Degunking Linux	18
Keeping It Simple	19
<i>Simplicity and Ease of Configuration</i>	19
<i>Resisting the Temptation of Complexity</i>	20
Keeping It Safe	23
<i>Murphy's Law</i>	23
<i>Attending to Security</i>	24
<i>Backing Up Your Data</i>	25
Eliminate Gunk	26
<i>Degunking User Files</i>	26
<i>System Software Degunking Strategies</i>	30
<i>Degunking Hardware and Drivers</i>	34

Additional Sources of Information	37
The Degunking 12-Step Program	39
Summing Up: A Mindset for Degunking	40

Chapter 3

Degunking User Files 41

Track Your Disk Usage	42
<i>Tracking Overall Disk Usage</i>	42
<i>Tracking Disk Usage in Specific Directories</i>	45
<i>Monitoring Your Disk Usage with a GUI Tool</i>	46
Identify, Delete, and Organize Your Files	48
<i>Identifying Files by Their Extensions</i>	50
<i>Identifying Files Types Using file</i>	52
<i>Finding Clues about Your Files Using strings</i>	53
<i>Other Methods of Identifying Files</i>	54
<i>Searching for Files</i>	55
<i>Organizing Your Files</i>	59
<i>When to Delete Unused Files</i>	61
Clean Up Dot Files	62
<i>What Are All Those Dot Files, Anyway?</i>	63
<i>Should You Delete Dot Files?</i>	66
<i>Methods of (Relatively) Safely Deleting Dot Files</i>	67
Summing Up: Cleaning Your Home Directory	68

Chapter 4

Degunking Your Desktop Environment 69

Tweak GNOME for Better Performance	70
<i>A Tour of Adjustable GNOME Features</i>	70
<i>Tweaking File Browsing in GNOME</i>	74
<i>Keeping Your GNOME Desktop Uncluttered</i>	78
<i>Starting with a Fresh GNOME Configuration</i>	78
Tweak KDE for Better Performance	79
<i>A Tour of Adjustable KDE Features</i>	79
<i>Tweaking File Browsing in KDE</i>	83
<i>Keeping Your KDE Desktop Uncluttered</i>	84
<i>Starting with a Fresh KDE Configuration</i>	85
Use Alternative Desktop Environments	85
<i>Using XFce</i>	86
<i>Using a Bare Window Manager</i>	89
<i>Rolling Your Own Environment</i>	92
<i>Configuring Linux to Use Your Alternative</i>	93
Miscellaneous Tricks for Improving a Linux GUI	97
Summing Up: Keeping Your Desktop Environment Gunk-Free	98

Chapter 5**User Settings for Common Applications 99**

- Degunk OpenOffice.org 100
 - OpenOffice.org Installation Options* 100
 - Managing OpenOffice.org Fonts* 104
 - Managing OpenOffice.org Printers* 107
 - File Compatibility Issues* 108
 - Faster Alternatives to OpenOffice.org* 109
- Degunking Mozilla Firefox 110
 - Improving Firefox's Fonts* 111
 - Improving Firefox's Security* 113
 - Alternatives to Firefox* 119
- Degunking Evolution 120
 - Reducing Clutter in Evolution* 121
 - Alternatives to Evolution* 126
- Summing Up: Keeping Gunk Out of Your Applications 127

Chapter 6**Cleaning Out Unused Packages 129**

- Why Clean Out Unused Packages? 130
 - Reducing Disk Space Consumption* 131
 - Maintaining Administration Simplicity* 131
 - Reducing Security Risks* 132
 - Minimizing Upgrade Headaches* 132
- Use Package Management Tools 133
 - Understanding Linux Package Systems* 133
 - Options for Package Browsers* 138
 - Using Red Hat Package Management* 138
 - Using Synaptic* 141
- Identify Unused Packages 143
 - Interpreting Package Descriptions* 143
 - Finding Additional Information* 144
 - Testing the Package* 145
- Keep Software Up-to-Date 146
 - Why Keep Software Up-to-Date?* 147
 - Using Update Agent for Software Updates* 148
 - Using Synaptic for Software Updates* 150
- Handle Dependency Problems 152
 - Tracking Down Dependencies* 152
 - Bypassing Dependencies: At Your Own Risk* 153
- Summing Up: Removing Package Gunk 154

Chapter 7

Improving Software Performance 155

- What Causes Sluggish Performance? 156
 - Software Problems* 156
 - Configuration Issues* 158
 - Interference Issues* 160
 - Hardware Issues* 161
- Improve Program Stability 164
 - Why Do Programs Crash?* 164
 - Working around a Crash* 166
 - Reporting Bugs* 167
- Improve Printing Performance 168
 - Understanding Linux Printing* 168
 - Degunking Your Printer* 171
 - Picking a Printer for Linux* 179
- Summing Up: Keeping Your System Zippy and Stable 181

Chapter 8

Managing Processes 183

- Understand the Linux Startup Procedure 184
 - The Tangled Web of Startup* 184
 - Intervening in the Startup Process* 188
- Locate Misbehaving Programs 193
 - Finding CPU Hogs* 194
 - Finding Memory Hogs* 197
 - Finding Processes by Name* 198
- Kill Stray Processes 199
 - Killing Processes Using top or GUI Tools* 199
 - Using the **kill** and **killall** Commands* 201
- Adjust Process Priorities 202
 - What Is a Process Priority?* 202
 - Identifying a Process's Priority* 203
 - Starting Processes with Non-Standard Priorities* 204
 - Modifying a Running Process's Priority* 205
- Summing Up: Keeping Processes in Check 207

Chapter 9

Account Degunking 209

- Why Use Accounts? 210
- Common Accounts That Are and Are Not Needed 212
 - Perusing Password Files* 212
 - System Administration Accounts* 214

<i>Special System Accounts</i>	215
<i>Accounts for Servers</i>	217
<i>User Accounts</i>	218
Create and Delete Accounts	219
<i>Managing Accounts with GUI Tools</i>	219
<i>Using useradd</i>	222
<i>Using userdel</i>	224
<i>Deleting a User's Files</i>	224
Set Passwords	226
<i>Selecting a Good Password</i>	226
<i>Setting Passwords with GUI Tools</i>	229
<i>Using passwd</i>	230
<i>Protecting Passwords</i>	230
Summing Up: Keeping Accounts Clean and Secure	233

Chapter 10

Network Degunking 235

Degunk the Web	236
<i>Gunk on the Web</i>	236
<i>Using a Degunking Web Proxy</i>	238
Clean Up Your E-Mail	244
<i>Using Spam and Worm Blockers</i>	244
<i>Reading Weird-Looking E-Mail</i>	250
Improve Your Network Security	252
<i>Removing Unnecessary Servers</i>	252
<i>Blocking Ports from Unauthorized Use</i>	255
<i>Using Encryption</i>	258
<i>Monitoring for Intrusion</i>	259
Summing Up: Keeping Network Gunk Off of Your Computer	262

Chapter 11

Finding Drivers for Your Hardware 263

Hardware Troubleshooting	264
<i>General Hardware Troubleshooting Procedures</i>	264
<i>Common Hardware Problems</i>	266
Find Sources for Hardware Drivers	271
<i>The Linux Kernel</i>	271
<i>Third-Party Kernel Modules</i>	275
<i>Hardware-Specific Software</i>	276
Install Drivers	277
<i>Installing Kernel Drivers</i>	277
<i>Installing X Drivers</i>	282

<i>Installing Printer Drivers</i>	284
<i>Installing Scanner Drivers</i>	284
Use the Last Resort: Replace Hardware	286
<i>When to Replace Hardware</i>	286
<i>Identifying Linux-Compatible Hardware</i>	287
Summing Up: Degunking the Hardware/Software Interface	289

Chapter 12

Optimizing Your X Configuration 291

The Role of X	292
Pick the Right X Driver	294
Improve X Performance and Appearance	295
<i>Setting the Color Depth</i>	295
<i>Setting the Resolution</i>	297
<i>Setting the Refresh Rate</i>	299
<i>Enabling 3D Optimizations</i>	300
Configure X Fonts	302
<i>Understanding Linux Fonts</i>	303
<i>Obtaining Good Fonts</i>	303
<i>Configuring X Core Fonts</i>	305
<i>Configuring Xft Fonts</i>	308
<i>Managing Fonts</i>	310
Summing Up: Keeping Your Display Clean	311

Appendix A

Backing Up Linux 312

Index 321

Introduction

When *Degunking Windows* first came out, I heard some people snidely comment that such a book was needed *only* for Windows. I wasn't so sure, though. Certainly Linux doesn't have a Registry to become gunked up, e-mail worms aren't running rampant among Linux systems, and Linux filesystems are much more resistant to speed degradation than are Windows filesystems, but Linux isn't without its problems. Unwanted files can easily accumulate in either operating system (OS). Linux has its own unique security challenges, and a Linux system can become bloated with unnecessary software, among other issues. The snobbishness of some in the Linux community aside, Linux can collect its own kinds of gunk, and this book is dedicated to cleaning it up.

Fortunately, Linux can be degunked. In fact, once you know how, Linux can be fine-tuned and work better than it did when you first installed it! This is partly due to the fact that Linux is a highly modular OS—almost every part of it can be replaced by a similar tool, enabling you, like Goldilocks, to pick the one that's just right for you, not too big or too small, too hot or too cold. This customizability, although it can be a great strength, can also lead to confusion and, realistically, gunk. The wrong choices can cause the system to perform poorly, not because the software is bad per se, but because it's just a bad fit for you. This book will help you find the right fit, as well as help you clean up your user files, configure the software you use in an optimal way, and otherwise clean up your system.

Who Should Buy This Book

I've written this book with a typical desktop computer user in mind. I expect you to understand the basics of how to use Linux and to have already installed the OS. I don't expect you to be a Linux geek, though—you don't need to be able to write SysV startup scripts from scratch, or even know what they are, for instance. Some Linux degunking tasks do invariably require delving into such issues, but when that happens, I try to do so in a gentle way.

This book provides practical advice on how to degunk your Linux system. It does *not* provide complete technical details of every command it describes; instead, it summarizes how to use programs to achieve specific goals, such as managing fonts or dealing with programs that are hogging your CPU time. This book also describes ways you can clean up your desktop, pick programs that will be best for *you*, and otherwise fine-tune your system.

Linux is unusual in that it's not a single OS; rather, it's a family of OSs. Each specific OS is known as a *distribution*—a collection of the Linux kernel together with support programs, an X server, startup scripts, and so on. The choices for each component, as well as the occasional distribution-specific tool, give each distribution its own unique “flavor.” Unfortunately, the plethora of available Linux distributions makes for problems when writing about Linux—the differences can make a statement that's valid for one distribution completely wrong for another. This book is written with all common desktop Linux distributions in mind, but it focuses most strongly on a few, including Debian, Fedora, Mandrake, Red Hat, and SuSE. In a couple of cases, I use tools that are specific to Fedora and Red Hat as examples, but other distributions' tools work in a similar way. If you're using Gentoo, Linspire, Slackware, Xandros, or something more exotic, you can still use this book, but you may run into some features that don't work in quite the way I describe. If so, you'll have to consult your distribution's documentation.

How to Use This Book

This book is structured in 4 parts with a total of 12 chapters, plus 1 appendix. Each part covers a particular type of degunking task:

- ✓ **Part I, Degunking Basics**—This part has two chapters that introduce the nature of Linux gunk and generalized degunking strategies. These chapters present the least in the way of specific degunking tips, but they're important for understanding the nature of Linux gunk. They're particularly valuable if you're relatively new to Linux, because they can help you orient yourself to the OS.
- ✓ **Part II, Degunking User Environments**—This part consists of three chapters that describe how to degunk user files, desktop environments, and common applications. For the most part, you can perform these degunking tasks as an ordinary user rather than as root, the system administrator.
- ✓ **Part III, System Degunking**—This part is the largest one in the book, at five chapters. It describes how to degunk the Linux system rather than user environments. Its chapters cover package maintenance, software performance, process management, account degunking, and network degunking. Most of the tasks described in this part of the book require root access.

- ✓ **Part IV, Hardware Degunking**—The final part of the book describes hardware-related degunking tasks—managing drivers and optimizing your X configuration. In some sense, these are specialized system degunking tasks; however, their close ties to the hardware create special considerations, such as the need to deal with the Linux kernel.

You can use this book in either of two ways:

- ✓ If you're having a specific problem, go to the chapter that seems relevant or look up the problem in the index. You should then be able to degunk that problem directly.
- ✓ If you want to degunk Linux generally, you should read through the book front to back. Generally speaking, the book progresses from simpler degunking tasks to more advanced degunking tasks.

Conventions Used in This Book

Writing about computers, and particularly about Linux, invariably introduces a certain amount of computer-specific jargon, some of which can be hard to distinguish from more common uses of some words. For this reason, this book uses certain typographical conventions to help clear things up:

- ✓ Normal text, like this, is used for the bulk of the words in most paragraphs.
- ✓ **Bold text** denotes uniform resource locators (URLs), which usually point to Web pages, as in **<http://www.paraglyphpress.com>**.
- ✓ Monospaced text denotes computer filenames, commands, and usernames. It's also used (mostly on lines by itself) to show the contents of Linux configuration files or text-mode interactions with the computer. This text may be *italicized* to indicate a variable—a word that you should change for your system or application.
- ✓ **Bold monospaced text** denotes commands that you type at a Linux command prompt.

Sometimes I show multi-line computer interactions. When doing so, I include a Linux prompt, at which you type commands. This prompt is usually either a dollar sign (\$), which denotes an interaction that you can perform as a normal user, or a hash mark (#), which denotes a command you must type as `root`.

Several multi-line elements appear in this book:

NOTE: A note provides additional information on a topic. This is information that's not critical to the main flow of the text but that could be interesting or important in certain contexts.

TIP: *A tip relays information that can be helpful in saving time or improving a configuration under special circumstances or that's otherwise particularly useful.*

WARNING! *Some commands and procedures, if used inappropriately, can cause real problems, and a warning tells you about these risks.*



GunkBuster's Notebook

The GunkBuster's Notebook is a special type of sidebar that provides real-world advice on application of the chapter's techniques or elaboration on a point to help take you further.

Attention to these typographical conventions will help you get the most from this book.



Why Is There Gunk in Linux?

Degunking Checklist:

- ✓ Learn how to quickly recognize Linux gunk.
- ✓ Learn why some things that users consider gunk on one Linux installation might be valuable for another Linux installation.
- ✓ Understand that there are three categories of gunk: user configuration gunk, system configuration gunk, and hardware gunk.
- ✓ Learn about the three ways gunk can be introduced into a Linux system: it can come with the distribution, it can be added by mistake, and it can be created maliciously.
- ✓ Understand the basic principles of degunking Linux.

Crack open a dictionary and you'll find that *gunk* is defined as a thick, greasy, unpleasant substance. This is the kind of stuff that would make you scream if it got on your clothes. Of course, computer software and configuration settings are virtual and thus cannot be *literally* thick or greasy. They can, however, certainly be unpleasant, and computer problems can definitely *seem* thick and greasy. Just think about the last time you had to stay up all night trying to solve a Linux configuration problem. You probably felt like you were stuck in the middle of a big gooey mess! You probably could have avoided the problem in the first place if you knew how to degunk your system.

One of Linux's strengths as an operating system (OS) is that it's not susceptible to many of the types of gunk that can plague an OS like Windows. If you've used both Windows and Linux, you probably realized this right away. Linux has no Registry—a gunk collector that merits a full chapter in *Degunking Windows*. Worms and viruses have yet to make any inroads in the Linux world. Nonetheless, Linux isn't immune to gunk; it just collects different *types* of gunk. In this chapter you'll learn about the types of gunk that can impact a Linux system. You might be surprised to learn about the types of gunk that can get in your way and slow your system down. The next chapter covers general techniques for dealing with gunk. Once you read Chapters 1 and 2, you'll have a good foundation for all of the degunking tasks that this book covers.

My goal is to help you focus on performing tasks to help you set up and customize your system so that you can save time and avoid headaches. You'll be amazed at what you can accomplish with a little degunking effort. As you'll learn, gunk is something that builds up over time and gets worse if you simply ignore it. The techniques I'll present are designed to help you get your computer to run as best it can, and if you follow the maintenance tasks I'll introduce, you'll keep everything working well.

NOTE: Technically, Linux is an operating system **kernel**—the core low-level program that interfaces to the computer hardware, manages memory, and provides other necessary services for regular programs. In popular use, the word **Linux** frequently refers to an entire OS—the kernel plus a plethora of tools and utilities that rely on it. In the Linux world, many ways exist to build an entire working OS atop a kernel, and many Linux **distributions** exist, each one a specific implementation. In this book, I write about Linux as an OS in general rather than about a specific distribution. Sometimes I need to refer to specific distributions, in which case, I identify them by name, such as Fedora Linux or SuSE Linux.

Learn How to Recognize Linux Gunk

To best deal with any problem, you must be able to first recognize it. Thus, before you begin degunking, you must be able to recognize gunk. In a computer

environment, this can get tricky—files you don't recognize might be the worst kind of gunk or they might be critical to your system's operation. Worse in some ways, something might be considered gunk on one computer but it could be a vital system component on another. For instance, you probably don't want to run a Web server program on a desktop computer, but that same program is the whole point of a Web server computer. It's very important for you to be good at detecting the difference because you don't want to remove or change something on a Linux installation and end up regretting it.

What Is Gunk?

I'll be using the term *gunk* to refer to any file or configuration that keeps a computer, user account, or any other subsystem on the computer from functioning at peak performance. A gunk-free computer will perform as well as it possibly can, given the limitations of how its hardware and software are designed. A gunky computer, on the other hand, will run slowly, will damage data files, won't perform some function that it should, or will otherwise behave in any way that makes you want to scream, rip your hair out, or toss the computer off the observation deck of the Empire State Building. The worst part about having a gunked-up computer is that it will waste your time and valuable resources in ways that you might not realize.

The challenge is that you may not always be able to tell what's gunky, simply because you might not know when something is working as well as it should. This is particularly true if you don't have a lot of experience with Linux. For example, you might not be able to answer questions like the following about your Linux setup:

- ✓ Is a 5-minute boot time for your computer slow, normal, or fast? (On most hardware and under most circumstances, it's slow.)
- ✓ Is it normal to have to use the root account for tasks like reconfiguring your network connection? (Yes, at least on non-dialup networks.)
- ✓ Is it normal to have to reinstall the same fonts multiple times in order to use them in all applications? (Sadly, the answer is often yes, although it depends on the applications.)
- ✓ Should a crashing program take down the entire computer? (Not normally; such behavior is often a sign of hardware problems.)
- ✓ Should users be able to read and write each other's files? (It depends on permissions on files and directories, which you can set as you see fit.)

As you spend more time with your Linux system and go through the process of degunking your computer, you'll learn to spot sources of gunk that can really

get in your way. I'll be providing you with an ample supply of pointers and steps so that you can quickly deal with the gunk that needs to be fixed.


GunkBuster's Notebook: One Person's Gunk Is Another Person's Treasure

One particularly tricky challenge to spotting gunk is that something that might be unwanted on one Linux installation might really be needed on another system. This might seem like a paradox, but the reason is that Linux can be used very differently by different users and Linux supports a variety of applications, from desktop systems to Web servers. Don't worry, though; throughout this book I'll point out situations in which you'll need to consider how your system is being used before you jump in and perform a particular degunking task.

Consider an example: Unnecessarily running a Web server opens the computer to outside access, and if the Web server is misconfigured or has a security problem, outsiders may be able to break into the system. On the other hand, if you *want* to run a Web server under Linux, the Web server software and all of its associated configuration files won't be gunk at all. You're also more likely to carefully configure and monitor a Web server program that you intend to run than one that's running accidentally (that is, a gunky Web server).

Another example of the relative nature of Linux gunk relates to choosing software for your system. In many respects, Linux is much more flexible than Windows. Linux provides more choices for many key software components, such as desktop environments, than Windows. This level of choice means that you can pick the software that's best suited to your specific needs, but it also means that an incorrect choice could be considered gunk. For instance, on a low-memory system, you should run memory-efficient programs rather than huge memory-sucking programs. On the other hand, these memory-sucking programs might provide desirable features—at least on systems with enough resources to spare.

In the end, you must be the ultimate arbiter of what is gunk and what is not. I can point out various choices, and describe the consequences of the choices, but you must make the decisions depending on how you are using your system (as a word processor, as a development platform, as a Web server, and so on).



You'll also need to take into account the capabilities of your hardware (a state-of-the-art 64-bit powerhouse, a 10-year-old system you can't quite bear to put out of its misery, or something in-between). Throughout this book, I'll describe how particular programs and configurations can be considered gunk or non-gunk in certain contexts, so I'll be there to help you figure things out. In the end, though, you must know your own system and make your own decisions about it.

You should also know that your own definition of gunk can change. If you try one program or configuration and then outgrow it, you can try another program or configuration. Over the long haul, this is inevitable—programs are updated, hardware improves, and you learn more. These factors can all lead you to new and improved configurations or make an old one look increasingly greasy and unpleasant.

Understand the Common Types of Linux Gunk

For this book, I've come up with three categories of gunk: user configuration gunk, system configuration gunk, and gunk relating to hardware or drivers. These categories mirror important distinctions in Linux itself. As a multiuser OS, Linux draws a fairly strong distinction between user and system configurations. User configurations apply to individual users. Two users can have very different user configurations without affecting each other. System configurations, by contrast, apply to everybody. Finally, hardware and driver configurations are, in some sense, system configurations, but they apply more directly to the hardware than do other system configurations.

Gunky User Configurations

User gunk is the stuff you must deal with as an individual user. If your computer has two users, one may experience problems but the other user won't. This type of gunk can also usually be cleaned up by the user who's affected by it, although there are some partial exceptions to this rule as you'll learn in this book.

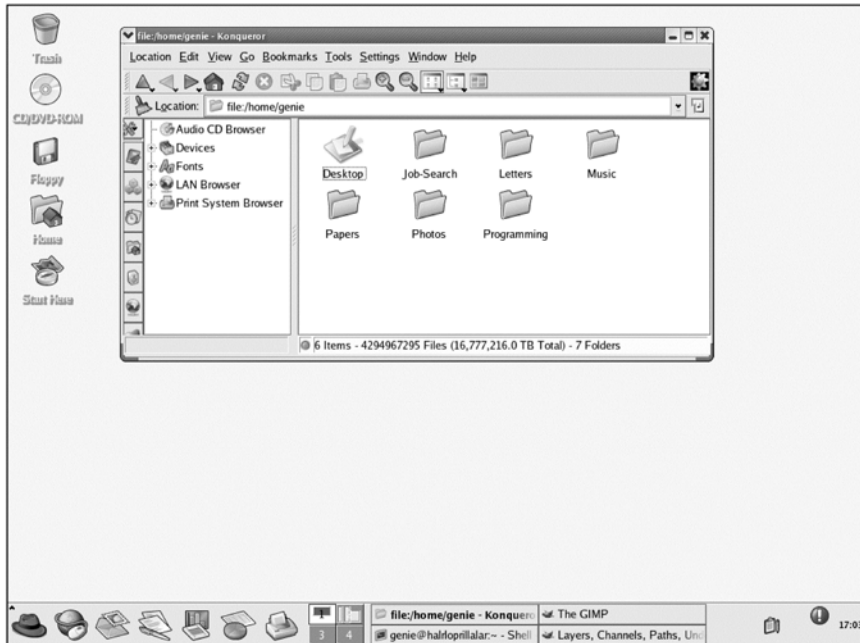
One common type of user gunk is unnecessary user files. Just like clutter in your home or office, unnecessary files can build up and clutter your user account. These files can eat up valuable disk space and make it difficult for you to locate files you do need when you need them most. You've likely already experienced

the frustration of using a Linux system in which you've had difficulty locating files. The easy solution is to spend the time you need to organize your files. Knowing when to delete files is also vitally important, but this usually requires knowing what the files do or what sort of data they contain, and that's not always obvious, particularly for Linux *dot files*.

Dot files have names that begin with dots (.). Most dot files are configuration files, and most tools hide them from view so they don't clutter your directory listings. Nonetheless, they do consume disk space, and because they are hidden, you might not even be aware of how they can be building up on your system. More important, they affect your user configurations—dot files hold defaults for your Web browser, e-mail reader, word processor, file browser, and so on. They're usually created automatically by the programs that use them, so if you try out a bunch of programs, your account will soon be crawling with dot files, and if you never use the programs they manage again, those files immediately become gunk. In Chapter 3, I'll show you how to clean up all of the dot files that you don't need.

Dot files can build up on your system, but an even bigger problem is that they can contain configuration settings that can gunk up your system. These configurations could be suboptimal or they just might not work. To configure some programs, you must directly edit their dot files, but the format of dot files isn't standardized. Thus, you'll need to learn about each one individually if you want to fine-tune them. Fortunately, most of the more modern GUI programs can also be configured through point-and-click interfaces. When you're done changing your configuration, the program silently saves its configuration in a dot file, so you needn't be concerned about the format of that file. Still, user program configurations can be gunky. They might not use the best fonts, or they might pose security risks, for instance. Although covering every single user program is beyond the scope of this book (it would really clutter up the book and make it gunky!), Chapters 4 and 5 do cover some of the more important and gunk-collecting programs.

Figures 1-1 and 1-2 illustrate the effect of some common types of user gunk. Figure 1-1 shows a relatively gunk-free desktop on a Fedora Linux system. This configuration uses the default desktop configuration and shows a file browser window open on the home directory, which contains subdirectories to hold specific types of files by logical categories. Figure 1-2 shows a much more cluttered display—it's populated with so many program icons that finding the right one is virtually impossible, and the program links and files in the home directory are in an unorganized mess, making it difficult to locate the right one. As this shows, it's very easy to let a user environment or account drift into chaos. Hopefully, the one shown in Figure 1-2 doesn't look like yours! If it

**Figure 1-1**

A gunk-free user environment tends to be simple and easy to navigate.

**Figure 1-2**

A gunky user environment is often cluttered and disorganized.

does, don't panic because this book can help you fight the tendency to accumulate clutter that can get in your way and reduce your productivity.

In an extreme case, you can start fresh by creating a new gunk-free user account for yourself—one at least free of the user configuration gunk that can collect over time. Unfortunately, this approach will also deny you access to any user-specific options you might have set, such as your Web browser bookmarks. In Chapter 5, you'll learn that there is a less radical way to deal with user configuration gunk.

Gunky System Configurations

A great deal of potential Linux gunk arises outside of user configurations. This system gunk frequently affects all of a computer's users because it drains overall system resources. It could also involve an unnecessary program that poses a critical security risk. Here are some examples:

- ✓ **Unnecessary software**—Unnecessary software can consume enough disk space to cause problems. If you accumulate too much software, you might even run out of hard disk space to do the work you need to do. With every system I've ever set up, I'm always confident that there is plenty of hard disk space to store everything. Over time, the free space just seems to evaporate, and all of the unnecessary programs certainly don't help. Some programs might even have similar names, which as you might guess, can cause a few headaches.
- ✓ **Sluggish software**—Some programs are just speedier than others. Sometimes the fix is to change programs, but other times you can improve the performance of a sluggish program in some other way. Occasionally, one program will go out of control and consume all your computer's CPU time, causing every other program to slow to a crawl.
- ✓ **Unreliable software**—Software that crashes or otherwise misbehaves is a real pain. Linux itself is a very stable OS, but given the huge number of programs available for Linux, it shouldn't be surprising that some programs have bugs, some of which cause the programs themselves to crash.
- ✓ **Account gunk**—Each time a new user account is set up on your computer, a bit of overhead is required. In addition, files stored in unused accounts can consume disk space. More important, unused accounts can provide intruders with a way into your system should they guess the password.
- ✓ **Network gunk**—In some sense, what I call *network gunk* goes beyond the system. This is gunk that originates on the Internet (or even a smaller local network) and that affects your computer. Examples include e-mail spam and worms, pop-up ads visible in Web browsers, and attacks from crackers on the

Internet. Defending against such gunk will help improve your overall network performance and might even prevent your system from being abused to send spam or launch attacks on other systems.

NOTE: The popular media refer to computer criminals as hackers. This word, though, has an older meaning: It refers to individuals who are skilled with computers, and particularly with computer programming, and who use these skills for productive and legal purposes. Many of the people who created Linux consider themselves hackers in this positive sense. For this reason, I use another term, *cracker*, to refer to computer miscreants.

GunkBuster's Notebook: Degunking with the Linux root Account

Degunking user accounts can get a little tricky because, for the most part, this activity requires access to the `root` account. (There are a few exceptions to this rule, though, or cases where the problem may straddle the line—`root` access might or might not be absolutely necessary to fix the problem in all cases.) Fixes usually involve modifying system configuration files or issuing commands that affect running programs. GUI tools are available for these tasks in many Linux distributions, but there's little cross-distribution standardization in GUI system administration tools. Command-line tools, though, work on all distributions, or at least on many of them. In this book, I'll emphasize the command-line tools, but I'll also present examples of GUI tool use for many tasks.

Linux separates user accounts from the `root` system administration account for a reason. Day-to-day use of Linux doesn't require access to files in `/etc` (the directory that holds most system configuration files) or other protected files, so using a normal user account helps protect against mistakes that could have dire consequences. Linux also provides few safeguards against errors when you use `root`. For instance, it's easy to accidentally delete all the files on a Linux system when using the `root` account, but this isn't possible when using a regular user account. Thus, you shouldn't use the `root` account unnecessarily, and when you do use this account, you should be particularly cautious about all the things you do with it. This distinction between system administration and ordinary use is a bit blurrier in Windows NT/200x/XP, and it's nonexistent in Windows 9x/Me. If you've moved from Windows to Linux, you might need some time to get used to this. You might even think the system of account management is a pointless hurdle to overcome, but the added security it provides is well worth the minor inconvenience of having to access the `root` account to perform certain tasks.

WARNING! *A few Linux distributions, such as Linspire (formerly known as Lindows), encourage use of the `root` account for everything. This configuration seems friendly to first-time Linux users, but it's very dangerous. I recommend avoiding such distributions, or at least reconfiguring them to use Linux accounts in the traditional way. A root-only configuration is not just ordinary gunk—it's biohazard gunk!*

Gunky Hardware and Drivers

So far you've learned that most Linux gunk consists of software gunk at a fairly high level—configuration files, user files, unnecessary programs, and so on. Your system can also be impacted by gunk that closely interacts with your hardware, or at least with your hardware drivers.

One type of hardware gunk is really quite severe: gunk that's so thick that it prevents your hardware from working. This gunk is typically a result of incorrect drivers. Most Linux drivers are part of the kernel, so you may need to learn a bit about the kernel to fix such problems. Some drivers reside in other subsystems, though. Most notably, printer, scanner, and most video card drivers belong to other programs. Knowing how to install and configure each of these driver types can greatly help you clean up this type of gunk. Sometimes, though, Linux simply doesn't support the hardware you're trying to use. In such cases, unfortunately, you may need to replace the hardware.

Hardware and driver gunk can be particularly troublesome because it can sometimes cause outright system crashes. When hardware is faulty or when a driver has a bug, it's likely that the problem will cause your computer to stop functioning altogether, not just slow down or behave erratically. Fortunately, such problems are rare, but they can be particularly vexing. Your system might crash randomly, giving you no immediate clue as to the cause. Chapter 11, "Finding Drivers for Your Hardware," provides a few tips on hardware troubleshooting.

One area of hardware that deserves special attention is video hardware. This hardware is controlled by the X Window System (or X for short), which is Linux's low-level GUI tool. X includes drivers for video hardware and handles low-level tasks such as creating windows and displaying text. Other programs expand X's operation, but basic X configuration deals mostly with X's interactions with hardware. Most distributions set these up in a reasonable way when they install themselves, but you may need to change these configurations to deal with new hardware or to fix a broken initial configuration. X's font handling is also notoriously tricky, although the situation has improved substantially in recent years.

Understand How Gunk Gets in Your System

You now should have some idea of the kinds of gunk your Linux system can collect. Before proceeding further, you should consider how the gunk got there. Knowing how gunk is created can help you develop successful defenses to block it in the first place—it's better to never have a problem than to have to correct one. Knowing how gunk is created can also help you develop a plan to cope with it. Throughout this book I will be showing you ways to get rid of the gunk you don't need, but I'll also point out ways that you can keep your system from collecting gunk in the first place.

Out-of-the-Box Gunk

One type of gunk is present from the start—it's installed along with Linux itself. Why would a Linux distribution developer place gunk in the distribution? The answer is that they don't know a thing about *your* computer or how *you* intend to use it and thus, they have to design their features to work with a variety of systems. This means that they'll likely provide stuff you don't really need. They can guess and develop distributions for particular *types* of users, but it's unlikely that they'll make guesses that are exactly right for *you*. When they guess wrong, the result can be gunk. Unfortunately, most Linux users simply get a distribution and install everything on the computer without giving the matter much thought.

Out-of-the-box gunk usually takes the form of too many installed programs, although poor defaults for desktop environments or other programs can be another form of this type of gunk. Each distribution has its own philosophy about what sorts of programs to install, so if you've not already installed Linux, you may want to consider the distribution's mix of packages, configuration tools, and default options to see which one is a good fit for you:

- ✓ **Debian GNU/Linux**—This distribution, headquartered at <http://www.debian.org>, takes a minimalist approach. A default installation provides very little in the way of user programs and other tools, although Debian provides lots of options to increase the software selection both during and after installation. Overall, a default Debian installation is very gunk-free, but to make Debian a useful distribution, you must manually install an assortment of packages, and knowing which packages to install can be tricky unless you're already familiar with Linux. In other words, Debian suffers from a sort of anti-gunk problem.
- ✓ **Fedora Linux**—Fedora (<http://fedora.redhat.com>) is the freely redistributable version of Red Hat and has the same advantages and disadvantages as that distribution.

- ✓ **Gentoo Linux**—Like Debian, Gentoo (<http://www.gentoo.org>) provides a rather minimalist (but gunk-free) experience from its initial installation, but this requires you to know what packages you want to install. Gentoo is unusual because its packaging system is built around compiling programs from source code. This can provide a modest speed boost once programs are running, but compiling programs from source is more time-consuming than installing them from pre-compiled binaries. If you are not a programmer or highly technical, you probably want to shy away from a distribution such as this.
- ✓ **Mandrake Linux**—This distribution is geared toward desktop users. It provides many useful user tools and emphasizes the K Desktop Environment (KDE) for its GUI, although you can use the GNU Network Object Model Environment (GNOME) instead, if you prefer. These characteristics mean that it installs a lot of programs that can easily be considered rather gunky, particularly if you prefer a more svelte system. Consult <http://www.mandrakelinux.com> for more information.
- ✓ **Red Hat Linux**—Red Hat (<http://www.redhat.com>) is one of the oldest current Linux distributions and is also a very popular one, particularly in North America. It comes in several varieties geared toward specific uses, from hobbyists (Fedora) to corporate servers (Red Hat Enterprise Linux). It uses GNOME by default, although you can opt to use KDE instead, if you prefer. In both cases, Red Hat makes more changes than is usual to the standard desktop environment. This doesn't really impact the amount of gunk it carries, but you might or might not like Red Hat's unique changes. Pick the right variant and installation options and Red Hat can be reasonably (but not completely) gunk-free, but pick the wrong variant or options and the system will be loaded down with gunk until you clean it out.
- ✓ **Slackware**—This distribution is the oldest of the surviving major Linux distributions. It qualifies as a fairly gunk-free distribution upon installation; like Debian and Gentoo, it tends to install relatively little in the way of extraneous doodads by default. Also like Debian and Gentoo, this very fact can make it somewhat unfriendly to those who aren't already familiar with Linux. Slackware ships with relatively little in the way of extra software, which helps minimize the chance of your unknowingly increasing the system's gunkiness, but at the expense of reduced flexibility. (You can always install software from other sources, though.) Consult <http://www.slackware.com> for more information.
- ✓ **SuSE**—SuSE (<http://www.suse.com> or <http://www.novell.com>) originated as a German distribution, but the company has since been purchased by U.S.-based Novell. This distribution is similar to Mandrake or Red Hat in the amount of gunk it introduces. The SuSE software selection is unusually broad, which can increase its level of gunk if you install everything—but this isn't the case by default, so the wide software selection is a plus.

- ✓ **Xandros**—This distribution is derived from Debian but has a strong desktop user emphasis. As such, it provides numerous tools that are handy for typical home and office desktop users, and this is nice if you happen to like the Xandros choices, which are built around KDE. If not, or if you want to run a server, these packages are gunk. If you want options not present in the standard Xandros system, you can install Debian packages, but doing so is more of a hassle than using the standard Xandros tools. Check **<http://www.xandros.com>** for more information.

Of course, this distribution list is far from complete; dozens of obscure Linux distributions exist, which install varying levels and types of gunk by default. Also, keep in mind that most distributions provide installation options that change the type of gunk they install. Most let you select or deselect individual packages or package groups during installation. Doing this on a package group basis can be handy because you can install (or choose not to install) an entire group of related packages with a single mouse click or keypress. This practice, though, increases the odds that you'll install at least some gunk. Perhaps you really need one package in a group, but not another. If you're very concerned about maintaining a gunk-free system, you might want to look over your choices carefully during system installation. If you've never used Linux before, though, you should probably stick with the defaults and then consult Chapter 6, "Cleaning Out Unused Packages," for information on degunking your package installations.

All distributions have at least minimal tools for detecting your hardware configuration during installation. These tools usually work fairly well, but sometimes they fail for one or more devices. This is particularly likely to happen when you use very recent or very exotic hardware. The result of hardware detection failures is often hardware driver gunk—the wrong driver loads, or a configuration that's intended to activate a device simply doesn't work. I'll be covering these in Part IV of this book.

One source of hardware gunk is common to all distributions: the default Linux kernel. The kernel interacts directly with most system hardware and is a necessary part of any Linux system. To work with a very wide array of computers, though, distribution developers must compile a fairly generic kernel. This kernel may contain drivers you don't need or use compiler optimizations that don't get the best performance out of your hardware. Recompiling your kernel can help clean out some of this gunk; however, this task requires extensive knowledge of your own hardware. Chapter 11 touches upon this subject but does not cover every detail of kernel recompilation. Think of it as an advanced degunking technique that requires special skills.

Gunk Added by Mistake

A second major way for gunk to make its way onto your system is through error. Errors can cause any of the types of gunk just described—user gunk, system gunk, or hardware/driver gunk. In fact, most user gunk is created in this way—Linux distributions provide relatively minimal user configurations. The configurations that they do provide are mostly centered around their desktop environments, and these usually aren't very gunky—at least, assuming that you want to use the desktop environment in question. (Big desktop environments can themselves be considered a type of gunk on small systems, as described in Chapter 4.) User errors that lead to gunk are largely housekeeping issues—a failure to delete old files or organize them into subdirectories, poor structure to icons added to a desktop environment, and so on.

System gunk and hardware/driver gunk that's added by mistake are usually the responsibility of the system administrator. On a home system or some small business systems, this may be the same person as the end user, but the actions that create the gunk are performed with the help of the root account. Frequently, this type of gunk is added because of inexperience. For instance, if you don't know much about certain packages, you might install more than you need because you want to install everything you *might* need. This tendency is understandable, but it does lead to gunk buildup, particularly over time. Similarly, inexpert changes to system configuration files (most of which are located in the `/etc` directory) can produce gunky configurations. When wearing the system administrator hat, your main defense against adding gunk by mistake is education. The more you know about how Linux operates, the less likely it is that you'll add gunk by mistake.

Gunk Added Maliciously

Perhaps the most troubling category of gunk is gunk that's added maliciously. Most typically, this is the result of a cracker who breaks into your computer. Sometimes this is done for fun, like a cyberspace joyride. Other times, the cracker has a specific goal in mind, and your system is just one stepping stone to that goal. For instance, crackers may want to use your computer to send spam or to attack other computers. On rare occasion, crackers break into your computer because you personally are (or your organization is) the target and the cracker wants to steal data, deface your Web site, or what have you.

In any event, crackers usually leave gunk behind. This gunk can take one or more of several forms:

- ✓ **Gunky accounts**—Crackers usually add accounts for themselves or modify existing accounts. This change enables the cracker to get back into your system with ease.
- ✓ **Added or modified programs**—Crackers often add new programs or modify existing ones. For instance, crackers might change the `login` program (which processes text-mode logins) to record passwords and send them to themselves.
- ✓ **Unauthorized servers**—Crackers sometimes run servers to enable them to get back into your system. This class of gunk may also require an added program (the server software itself), but installing the software and running it are two different things. A more subtle change might involve running a server that was already installed but not active, or changing a server's configuration so that it accepts unfriendly connections you'd previously blocked.
- ✓ **Modified files**—If you were specifically targeted, user files might be modified, as in a defaced Web page. Most intrusions also involve modified configuration files. (Gunky accounts are really just modified account database files, for instance.)
- ✓ **Sloppy changes**—Many crackers have limited skills (they're often called *script kiddies*, the diminutive form reflecting the intruders' lack of real skills). These crackers use *root kits*, which are semi-automated attack programs written by others, to gain access to systems. Once in, though, script kiddies are often such poor administrators that they cause any number of problems through incompetent changes to your configuration, even if this wasn't their intention.

Unfortunately, gunk that's been added maliciously is tricky to clean up. The problem is that you can't be sure of the extent of the infestation. Perhaps you've spotted a bogus account in the account database, but that won't tell you whether the `login` program has been modified. Thus, the safest way to deal with this problem is to completely reinstall your system and to beef up security along the way. If you're lucky enough to know how the intruders got in, you should be able to fix the problem. If not, though, you may need to attend to security generally—in other words, clean up the gunk that came with the distribution or that you added yourself by mistake.

Learn How to Eliminate Linux Gunk

The preceding pages have alluded to several approaches to cleaning up gunk, and the next chapter provides many more tips. Broadly speaking, though, gunk-cleaning in Linux is handled in several ways:

- ✓ **Prevention.** Arguably, the best way to eliminate gunk is to keep it from fouling your computer to begin with. This begins with picking a relatively gunk-free distribution or installing Linux with few options. Researching packages before installing them, and installing only those that you really need, are additional important preventive measures. Likewise for performing system configurations—study the options and their effects rather than blindly applying features that sound like they *might* be useful. On the user side, keep your directory tree organized from the start and don't create files unnecessarily. When you do create files, delete them once they're no longer needed.
- ✓ **Correcting user problems**—User problems, once they've emerged, usually require cleaning out individual files in a user account or editing user configuration files. The latter task can usually be done in the program that created the configuration file, but occasionally you'll need to break out a text editor and use it on a dot file.
- ✓ **Correcting system and hardware/driver problems**—These problems have many different causes, and therefore many different solutions. You might need to remove or upgrade a program using your distribution's package-management tools, edit a system or program global configuration file, monitor processes for signs of problems, or perform various other tasks. As a general rule, this type of problem requires the most skill to correct. It's also the most dangerous type of gunk to clean—if you're not careful, you can damage the Linux system, just as you can damage your car's paint if you try to wax it before you wash it.

Summing Up: A Taxonomy of Gunk

Linux, although arguably a cleaner OS than Windows, has its share of problems—its own gunk, if you will. Although precisely what constitutes gunk depends on your point of view, any given item of gunk can be categorized as falling into one of several basic classes: user gunk, system gunk, and hardware/driver gunk. Understanding what these categories are, how they get onto a Linux system, and how they can be removed are the keys to keeping your system gunk-free.

As you'll learn in the next chapter, the most difficult part of degunking is simply setting some time aside to perform some cleaning tasks on a regular basis. Subsequent chapters present the tasks in the order that will likely give you the best results in the shortest amount of time.



Linux Degunking Strategies

Degunking Checklist:

- ✓ Keep your system configuration and user files as simple as possible.
- ✓ Understand that anything that creates an unsafe computing environment is the worst type of gunk.
- ✓ Learn the importance of setting up a good backup system as part of your overall degunking strategy.
- ✓ Understand the basic principles of how to eliminate gunk in Linux—user gunk, system software gunk, and troublesome hardware/driver gunk.
- ✓ Learn the degunking 12-step program.

In Chapter 1 you learned about the different types of gunk that can impact your Linux configuration—user gunk, system software gunk, and hardware/driver gunk. My goal now is to introduce you to important degunking strategies that can help you save time, can reduce aggravation, and can help you be more productive. Starting with Chapter 3, I'll be presenting very specific techniques for degunking and fine-tuning your installation of Linux. Before delving into these specifics, though, I'll present some important general procedures and techniques for degunking Linux. These principles apply to many of the specifics covered later in this book, as well as to many other types of Linux gunk.

The final goal of this chapter is to present the degunking 12-step program. These important 12 steps will provide you with the road map you need so that you'll be able to get the most out of your Linux configuration in the shortest amount of time. For those with a limited amount of time, the program is designed so that any step you perform will help you become more productive.

The Strategy for Degunking Linux

The strategy for degunking your Linux configuration is based on how Linux operates:

- ✓ How Linux's Unix heritage affects the programs available for Linux and the overall Linux/Unix philosophy for computing
- ✓ How Linux is designed as a basic operating system *kernel* but specific Linux distributions provide the kernel along with a plethora of tools and utilities
- ✓ How Linux stores certain types of files in specific directories
- ✓ How Linux distributions provide numerous configuration files that can help you customize your system
- ✓ How different Linux distributions provide different utilities and support files
- ✓ How the different Linux desktop environments introduce different levels of complexity
- ✓ How Linux supports hardware and peripherals
- ✓ How Linux is often used for heavy-duty roles such as Web servers, which introduce unique security risks
- ✓ How Linux supports different types of system and user accounts
- ✓ How Linux supports security features
- ✓ How different Linux distributions provide tools for backing up and recovering data

- ✓ How Linux's open source nature enables you to tweak the system, even modifying its component programs

As you'll learn in this book, the more you understand about how Linux is designed and how it operates, the better you'll get at degunking your particular setup. What makes degunking and fine-tuning Linux especially challenging is that there is no one standard that you'll encounter. Although Linux is designed around a core operating system kernel, each of the different distributions provides numerous options, utilities, and support programs.

Keeping It Simple

The first degunking strategy is to simplify your configuration. A simple configuration makes it easier to find and correct problems. Like a smooth countertop, a simple computer configuration is easy to clean compared to something with lots of decorative ridges. Complex configurations might seem appealing at first because they often provide features that tend to draw you in, but like the Sirens calling to Odysseus, they can lead to disaster, so resisting this call can be important. On the other hand, sometimes complex configurations really are necessary, or at least they provide enough benefit to make their extra risks or hassles worthwhile. Ultimately, you must be the judge of how complex your configuration should be. I recommend, as a general rule of thumb, starting simple and then building from there. That gives you time to learn about the tools you add to make a complex configuration, and that knowledge will be helpful in solving any problems that might arise.

Simplicity and Ease of Configuration

Every year, computers grow more complex. Even aside from the increasing complexity of CPUs (which is largely hidden from users), programs almost always acquire features rather than shed them, and operating systems (OSs) accumulate files and programs like dust bunnies under a bed. At worst, these new features consume resources and add complexity while providing little or no benefit in return. At best, these new features and programs add something useful to a subset of the target audience. Sometimes this subset is large, but frequently it's not. How many people need spell-check dictionaries for a dozen languages? How many people regularly run four desktop environments? For people who don't need all these features, the unused features are gunk. Unused features and programs pose several problems:

- ✓ **Wasted resources**—Unused features and programs take up disk space. They may also consume RAM and may even chew up CPU time. The result is an artificially inflated need for system resources. Computer manufacturers must love this because it drives sales, but for ordinary people, it's just gunk.

Fortunately, Linux is built in a modular way that makes it easier to slim down than many OSs, so you can degunk Linux enough to let it run well on hardware that's older or less capable than many modern competing OSs require.

- ✓ **Security threats**—Unused programs can pose a security threat. This is particularly true of server programs that are left running unnecessarily because bugs or misconfigured settings can enable outsiders to break into your computer. Unused features of programs can also be a security problem because every extra line of code increases the risk of a bug being introduced, and bugs sometimes spawn security problems.
- ✓ **Configuration effort**—Many programs require some effort to configure. You might need to edit a configuration file, run a configuration tool, or what have you. Some require ongoing monitoring. That said, some programs are designed to *ease* administrative burdens by helping to automate the process. When well designed, such tools can be useful; however, sometimes this strategy backfires. A poorly designed automation tool can actually create headaches, forcing you to disable or bypass it when it misconfigures a feature.
- ✓ **Learning curve**—Complex programs often take more effort to learn to use than simple programs. A new user who just wants to write a letter is often better off with a simple word processor such as Abi Word than with a complex one such as OpenOffice.org.
- ✓ **Daily use problems**—Complex programs can be a struggle to use for simple tasks. Wading through cluttered menus can hinder productivity rather than promote it, even if the features in those menus are intended as time-savers.

For all of these reasons, many programmers and system administrators adhere to the *Keep it Simple, Stupid (KISS)* principle, which holds that individual programs and overall system configurations should be kept as simple as possible. Doing so reduces the opportunity of gunk to gain a foothold on your system. Taken to the extreme, this can result in a very spare environment indeed. Most people today aren't comfortable with such environments, but it's possible to apply the KISS principle to a less radical extent. Doing so can help reduce your system's complexity without tying your hands in terms of helpful features.

Resisting the Temptation of Complexity

As a general rule when considering a program or configuration, you should ask yourself if adding the new tool will really help your productivity or enjoyment of the system. Most potential additions at least *could* be helpful, and it's easy to

fall into the trap of adding one potentially useful program or feature after another until your system is crawling with gunk. Try to think of every new addition as a tacky plastic pink flamingo that'll adorn your front lawn or a useless knick-knack that'll clutter your living room. Is that addition really necessary, or will it sit unused, collect dust, and generally become a hassle?

Sometimes you might not be able to tell if something is really necessary or helpful until you try it. Certainly there's nothing wrong with installing a program or trying a new feature to evaluate it. You should, though, keep at least a mental note of such additions and if they don't work out, remember to uninstall the software or, if applicable, disable the feature if you decide it's not worthwhile. (Chapter 6, "Cleaning Out Unused Packages," can help you identify and delete unused programs.)

With each new release of an OS, the software can become more complex. Linux distributions tend to collect a lot of gunk, although not all of it is installed on all systems. Nonetheless, the increasing size of the distribution as a whole is reflected in overall total increased size of the installed system. Similar comments apply to individual programs, such as mail readers and spreadsheets.

Fortunately, Linux follows a Unix philosophy of modular design. Many Linux programs and subsystems are actually composed of multiple smaller programs, each of which is fairly gunk-free. Frequently, these programs can be installed independently of one another. This modular philosophy enables you to design a Linux system that has very little gunk—you needn't install the packages that you don't really need, even if those packages provide features that would extend the capabilities of programs you do want to run. The downside to this design is that you need a great deal of knowledge of and experience with the programs in question to properly fine-tune the configuration.

For instance, consider the desktop environment—the GUI tools that greet you when you log into a Linux system that runs in GUI mode. The desktop environment runs atop the X Window System (or X for short), and Linux provides several choices of desktop environments. The GNU Network Object Model Environment (GNOME) and the K Desktop Environment (KDE) are the most popular choices, but others are available. GNOME, KDE, and most other desktop environments come in multiple packages, so you can install just those packages you need. You can even mix and match components—say, to run GNOME games from KDE.

Another approach to reducing complexity is to switch packages. If the word processor program you're using is growing too complex for your taste, investigate

alternatives. Chances are you'll be able to find a simpler one that will work as well for you but consume fewer system resources or otherwise be less gunky.

GunkBuster's Notebook: When Complexity Helps

The preceding description may make it sound as if complexity is the scourge of the software world, causing more problems than it solves. In reality, developers add features to their programs for reasons, most of which are perfectly valid. Complex programs can, after all, do more things for more people than simple programs can. This makes complex programs more appealing than simple ones, and sometimes complex programs are even necessary. Similarly, complex system configurations provide features that can be a great boon to users by providing useful features.

The trick, as always, is in finding the proper *balance* between complexity and simplicity. This task is tricky because the right balance varies from one system and user to another. In evaluating specific programs and configurations, you can ask yourself some questions to determine whether anything you might add will be beneficial or be gunk:

- ✓ **Do I really need the item?** If the program or configuration is literally essential for your use of the system, it's a necessary complexity, not gunk. A Web server program is obviously necessary on a computer that's to function as a Web server, for instance. You might even need a very complex Web server if your site must support complex functions provided by such programs.
- ✓ **If I don't really need the program or configuration, will it nonetheless help me?** A program or configuration can be non-gunk even if it's not strictly necessary. As noted earlier, some people like to work with very Spartan Linux configurations, but such setups aren't to most people's liking. Using X, GNOME, OpenOffice.org, and other large programs can be perfectly fine if you work better with such tools than with simpler counterparts. Even if they don't increase your productivity in any measurable way, you might simply *like* a certain configuration, such as looking at a favorite photo as desktop "wallpaper."
- ✓ **Do I have the hardware and software resources to run the program or configuration?** Although a setup might in some sense be gunky, that fact might not be

a real problem if your system is powerful enough. For instance, installing KDE on a system with 300GB of hard disk space isn't likely to be a big waste of disk space even if you never run KDE; the few megabytes consumed by KDE are simply trivial compared to the available disk space. On the other hand, making such decisions on a regular basis can add up quickly. Sooner or later, you should draw the line.

- ✓ **Is the software or configuration likely to cause problems in the future?** As a practical matter, gunk is gunk because it causes problems. An unnecessary or overly complex configuration may need extra maintenance, pose security risks, and so on, but some additions are less likely to cause such problems than others. An unnecessary word processor is less likely to become a security risk than is an unnecessary Web server, for instance.

Ultimately, you must decide for yourself whether or not something is gunk. Experience will be your best guide in the long run.

Keeping It Safe

Anything that creates an unsafe computing environment is the worst type of gunk. Safety can be an issue of programs that misbehave and lose data or programs that present security risks. Understanding when things are likely to go wrong and knowing how to attend to security issues are critical to keeping this worst type of gunk out of your system. Backing up your computer, or at least your most important data files, is also a vital safety precaution.

Murphy's Law

We all know the saying "If anything can go wrong, it will." This pessimistic appraisal is at the heart of safety planning because if you truly believe it, you'll do your best to ensure that nothing *can* go wrong. Could an intruder break into a little-used server program? If so, and if it's not vital, shut it down so that an intrusion via this route simply isn't possible. Could data be lost to a hard disk crash? If so, back up the data—and create a backup of the backup because the first backup might fail as well!

You should also consider that we can't always imagine all the possible points of failure. The result can be something going wrong that we didn't foresee. In terms of computer configuration, such lack of imagination is frequently the

result of lack of knowledge. Thus, you should learn as much as you can about Linux and about your Linux system in particular.

All this said, as a practical matter you must stop somewhere. Safety isn't absolute, it's relative. Even the most secure computers in the world can't be absolutely 100-percent secure. Chances are you're not configuring a CIA database with top-secret information, and you probably don't want to devote every waking hour to fussing over your computer. Fortunately, you can make a system *reasonably* safe by following some relatively simple procedures. In fact, a lot of this book is devoted to such procedures because a lot of gunk has safety implications.

Attending to Security

Entire books have been written about Linux security, but broadly speaking, the most important security steps that relate to degunking are as follows:

- ✓ **Physically secure the computer**—You should ensure that your computer isn't susceptible to physical attack. Lock it in a room, or at least chain it to a wall. If it's in a public place, lock the case shut. Just how much physical security is appropriate depends on the sensitivity of the computer and its intended use, of course.
- ✓ **Shut down unnecessary servers**—Server programs that don't need to be running should be shut down. These programs, by their very nature, listen for outside connections. Bugs in such programs can give crackers a foothold on your system. Chapter 6 will help you locate and eliminate such programs.
- ✓ **Eliminate unnecessary programs**—Local non-server programs can be a security risk should a user be a "bad apple" or should a cracker break into an ordinary account. This risk is usually quite minor, but a few programs have privileged access to the system, even when they're run by ordinary users. Thus, you should uninstall programs that aren't really necessary. Again, Chapter 6 describes this process.
- ✓ **Keep software up-to-date**—Software updates sometimes include security-related patches. Most Linux distributions provide tools to help automate software updates, so consult your distribution's documentation for details. Chapter 6 provides information on some popular software update tools, as well.
- ✓ **Delete unused accounts**—Linux supports multiple accounts, and these can accumulate over time. Chapter 9, "Account Degunking," provides information to help you with this task.
- ✓ **Monitor the system**—Being aware of the computer's operation can go a long way toward keeping it secure. Investigating peculiarities such as sudden slowdowns or programs starting to behave strangely can help a lot. In

extreme cases, you may want to install special monitoring tools, such as Tripwire (<http://www.tripwire.com>), to help spot intrusions. Note that this step is more reactive than preventive. The idea is to detect an intrusion early to minimize the damage.

- ✓ **Use good passwords**—Poor passwords can be a big security risk. They might be easy to discern from a distance as they're typed in a public or semipublic area, and they can be recovered should an intruder obtain a password file. You should also *never* share a password. Chapter 9 describes password strategies.
- ✓ **Log out of your account**—When you're done using your computer, log out of the account, or at least use a screen saver that requires a password to let you back in. This practice is particularly important if you use a computer in a public or semipublic area, such as a corporate “cubicle farm” or a university computing center.

Backing Up Your Data

Even the best care in keeping your system safe and gunk-free isn't a 100-percent guarantee against disaster. You can slip up and accidentally delete a file, a hard disk can fail, or your computer might be destroyed in a fire or other disaster. For such occasions, backups provide an invaluable means of recovery.

NOTE: *Backing up your computer can seem tedious, but in the long run, backup hardware and software can save far more effort than it imposes on you. This is particularly true if you use the computer for labor-intensive or mission-critical applications or if loss of data or system downtime will cause serious disruptions.*

Backups can range from a few files casually dropped on a floppy disk for safe-keeping to elaborate network backup plans that use a *backup server* to store data for an entire network. The type of backup you use should correspond with the importance and quantity of your data and the number of computers you want to back up. Broadly speaking, there are two types of backups you can perform:

- ✓ **User data backups**—Any backup of user data, as opposed to Linux system files, qualifies as a user data backup. This type of backup may be adequate for casual home users or for business workstations that have standardized system configurations and some means of replicating the basic installation. User data backups can be performed in a fairly informal manner (say, backing up files on an ad hoc basis to CD-Rs) or through a more formal backup plan (say, using a network backup server to do the job on a regular basis). They're usually fairly easy to restore.
- ✓ **System backups**—A more thorough type of backup stores every file from a computer on a backup medium. A system backup enables you to recover a

working system to a fresh disk or computer in a single step. This can be very helpful for mission-critical servers and the like—say, to speed recovery in case of a sudden hard disk failure. These backups require more planning than informal user data backups. A full restore can be trickier to implement, but restoring just some data from such a backup usually isn't too difficult.

These backup types aren't mutually exclusive, and in fact the line between them can be blurry at times. A complete system backup can be used to recover user data files, much like a user data backup, so if you do full system backups, you needn't perform separate user data backups.

NOTE: *Backing up your system is covered in more detail in Appendix A.*

Eliminate Gunk

In Chapter 1, you learned about the three major types of Linux gunk—user configurations, system software configurations, and hardware/driver gunk. These three categories are fairly distinct in Linux, although there are some areas of overlap. Nonetheless, each requires its own degunking strategy, as I describe next.

Degunking User Files

User configurations affect individual users—they define a user's desktop environment, hold a user's files, and so on. Thus, most gunky user configurations don't affect other users. (A single user might consume all the available user disk space, though.) Broadly speaking, user configuration gunk falls into two categories: gunky program configurations and cluttered files.

Degunking Program Configurations

Program configurations can accumulate program-specific gunk. Typically, this is a question of options you've set in the program that affect just one program. (Settings in KDE, GNOME, and other desktop environments can affect multiple programs, though.) Numerous things can go wrong:

- ✓ **Ugly fonts, colors, or styles**—You might set a program to use fonts, color schemes, or other stylistic features that you decide are ugly. In a worst-case scenario, you might configure a program to use a special font, which could make changing back difficult—after all, unless you've been studying at Starfleet Academy, you might have trouble finding the font option in your menus if you've accidentally changed your font to Klingon!

- ✓ **Altered menus or features**—Some programs provide options to increase or decrease the number of options on menus. This is particularly important for desktop environments, which provide these features to enable you to launch extra programs or unclutter your environment. A slipup might be difficult to correct, though—for instance, you might be unable to locate a program you’ve accidentally removed from a menu.
- ✓ **Too-complex configurations**—In keeping with the KISS principle, you might want to simplify a configuration, even if it’s the default one. Sometimes this can’t be done (say, if a program’s menus can’t be altered to eliminate unneeded options), but other times it can.
- ✓ **Flat-out broken configurations**—Sometimes a program’s configuration will be broken so badly that the program won’t work correctly. This is particularly common when you try to move a configuration from one Linux distribution to another—for instance, if you try replacing Red Hat with Debian on a computer while maintaining your existing home directory. Because these distributions place files in different places, you can end up with nonworking features, missing icons, and so on.

In theory, the best way to cope with most of these problems is to use the program’s own configuration tools. Look for a menu option called Preferences, Options, Settings, Configure, or something similar. Such an option is frequently, but not always, under the File or Edit menu in the program. The result is usually a dialog box in which you can change various program options. For instance, Figure 2-1 shows the Preferences dialog box from the GIMP, a popular Linux graphics program. Such configuration tools frequently provide a set of option categories (selected from the list in the left pane in Figure 2-1); each category provides its own set of options. Peruse this list until you find the option you want to modify then do so.

GunkBuster’s Notebook: Configuring Multiple Applications

Traditionally, each Linux program has used its own configuration file exclusively. For instance, setting the fonts used by a mail reader probably won’t affect the fonts used in any other programs. This design has its advantages and disadvantages—it enables fine-grained control over such features, but it makes it impossible to set features globally.

Increasingly, Linux programs are associated with desktop environments. KDE and GNOME both provide configuration tools that can set options for all the programs associated with their

projects. This tool is called the Control Center, but KDE's Control Center and GNOME's Control Center are independent entities. When you set options in a Control Center, those options affect all the affiliated programs. (Sometimes these programs provide overrides for some or all features, though.) For instance, you might set the default menu font in KDE's Control Center and this default will apply to all KDE programs. This change will not affect GNOME or unaffiliated programs, though.

NOTE: Efforts are underway to provide increased interprogram option integration. Thus, in the future, changes made to at least some features in Control Center and similar programs may apply to a much wider range of programs.

Unfortunately, degunking configurations using programs' own tools sometimes doesn't work. This is particularly true when a configuration is so badly broken that you can't find the relevant options, as when all the text appears in Klingon. For such situations, a more radical fix exists: Eliminate the user configuration files. These files are *dot files*—files whose names begin with dots (.). Typically, the dot files are named after the program they control, such as `.xcdroast` for the X-CD-Roast program. Sometimes, dot files are actually subdirectories, which hold multiple files, other times, they're truly individual files. In either

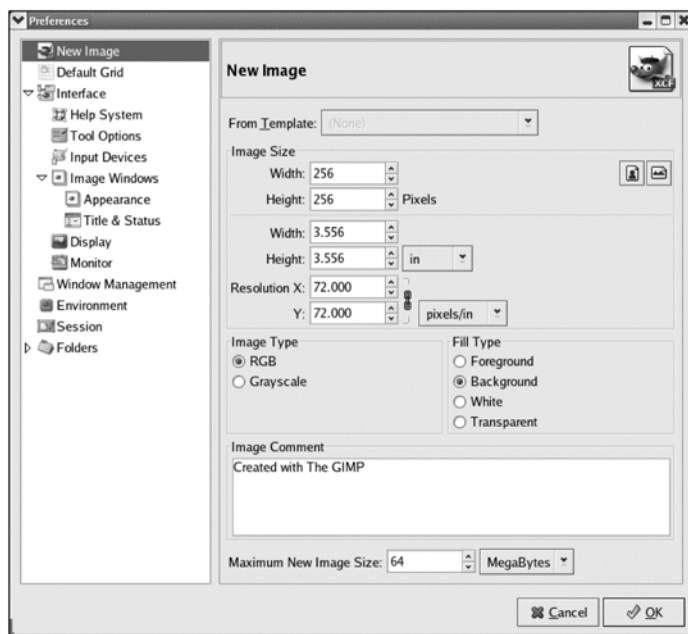


Figure 2-1

Many GUI programs provide a tool for setting program options.

case, if you can locate the correct dot file for a program, you can delete it. (I recommend renaming it instead of deleting it so that you can move it back into place if you located the wrong file.) When you next run the program in question, it won't be able to find the dot file and so will create a new one with the default settings. Chapter 3, "Degunking User Files," describes dot files in more detail, including how to locate and delete dot files.

Eliminating Clutter

A second class of user gunk is simple clutter. It's easy to create a very cluttered home directory, containing a mishmash of too many files and file types. The result is that it's hard to find the files you want—they all tend to blur together when you view them in a file listing. Two keys exist to eliminating clutter:

- ✓ **Create and use a logical directory structure**—Use Linux's subdirectories (often called folders, particularly when a GUI file manager is used). Create subdirectories for different file types or projects. For instance, you might create one subdirectory that holds papers you're writing, another for music files, and a third for financial data. Each of these might itself contain subdirectories, such as one subdirectory for each paper you're writing. Typically, though, you should only create a subdirectory if it will hold several files.
- ✓ **Regularly clean out unnecessary files**—Creating files is easy, and they tend to accumulate if left unchecked. This is a waste of disk space and also simply adds to the clutter, particularly if you don't maintain a good directory structure. Thus, you should periodically go through your files and delete those that aren't necessary. Chapter 3 provides much more detail on this task, including information on tools that can help you do it.

Newcomers to Linux are often unfamiliar with the Linux directory system. User files are mostly confined to users' home directories. These directories are typically located in `/home` and named after the users' accounts. For instance, the user `gerald` might have a home directory of `/home/gerald`. All of this user's files therefore reside in this one directory. Exceptions to this rule do exist, though. In particular, temporary storage areas such as `/tmp` can hold files owned by any user, and users can often mount removable media such as floppy disks in the `/media`, `/mnt`, or certain other locations and then store files there. Confining most user files to home directories simplifies overall system configuration, particularly on multiuser systems—a system administrator need not worry about deleting user files when deleting program directories prior to installing a new version of the program. If you've used Windows or other OSs in the past and are used to storing data files alongside programs, you should break yourself of that habit. (In fact, it's hard to indulge in this habit when using Linux!) Keeping your user files in your home directory will greatly simplify matters in the long run.

System Software Degunking Strategies

Recall that the second class of Linux gunk is system software gunk—gunk that’s associated with programs or system configurations as a whole. This gunk tends to affect all users, or at least all users who run certain programs or perform certain operations. Cleaning up this gunk usually requires root access. The first principle in system software degunking is to understand how the OS works. A basic tour of the Linux boot process and system configuration files will help you identify likely sources of system problems. Global program configurations, like individual users’ program configurations, can also be a source of problems, so you should know where to look for such problems. Finally, a lot of system software gunk is associated with *packages*—collections of programs and other files that can be installed and managed as a unit. Understanding a bit about Linux package management will take you far in dealing with package degunking.

Understanding System Operations

Chapter 8, “Managing Processes,” describes the Linux startup process in detail—how the kernel boots, what programs and scripts control the startup process, and so on. In brief, though, a program known as a *boot loader* loads the Linux kernel into memory from disk. The kernel then looks for key hardware components and configures them. When the kernel is through with this, it launches a program called *init*, which in turn launches a series of *SysV startup scripts*. These scripts start networking, launch servers, and otherwise perform critical system startup tasks. The *init* process also launches text-mode login prompts, enabling you to log into the computer. GUI login prompts may be launched through *init* or through a SysV script, depending on the distribution.

Because of *init*’s importance, you should know that it’s controlled through the */etc/inittab* file. For now, you should simply know not to mess with this file—incorrect changes to */etc/inittab* can render a system unbootable! The SysV startup scripts are also quite important. They’re usually located in */etc/rc.d*, */etc/init.d*, or */etc/init.d/rc.d*. (The precise location varies from one distribution to another.) Subdirectories of */etc* or of one of the main SysV script directories are named *rc?.d*, where *?* is a number from 0 to 6. These numbers refer to *runlevels*, which are sets of programs and services that can be started or stopped as a group. The runlevel subdirectories contain symbolic links that point to the real SysV startup script files. When a new runlevel is entered, Linux runs the SysV startup scripts to start or stop the appropriate services. In sum, the SysV startup script and runlevel directories control how most programs start when the system boots and also how they stop when the

system shuts down. Problems in the startup or shutdown procedure can often be traced to these scripts or their links. Chapter 8 describes this process in more detail.

You may have noticed that these critical startup scripts and configuration files all reside in `/etc`. This directory, and its subdirectories, contain most system-wide configuration files, including those for Linux as a whole and for many individual programs that have system-wide configuration files. Thus, `/etc` is a common destination when dealing with system configuration issues. It can also easily collect gunk—if you install a program and then later uninstall it, the program's configuration file may end up being left behind. This type of gunk is unlikely to do real damage, but it can clutter up the `/etc` directory unnecessarily.

A few files and subdirectories in `/etc` are particularly noteworthy:

- ✓ `/etc/X11`—This subdirectory holds configuration files for X and for a few important X programs. Most noteworthy among these files are `/etc/X11/XF86Config` and `/etc/X11/xorg.conf`, the configuration files for the XFree86 and X.org-X11 X servers, respectively. (X.org-X11 replaced XFree86 as the default X server on most Linux distributions in 2004.)
- ✓ `/etc/crontab`—This file controls `cron`, a Linux tool for running programs at regularly scheduled times. Related directories begin with the word `cron`, as in `/etc/cron.d`. Details vary from one distribution to another, though.
- ✓ `/etc/cups`—This subdirectory contains files that control the Common Unix Printing System (CUPS), which is used by most modern Linux systems for printing. CUPS is best configured through GUI tools, which edit the files in this subdirectory.
- ✓ `/etc/fstab`—This file is very important. It describes where Linux should mount various partitions and other disk devices. For instance, it might say to mount `/dev/hda7` (a particular disk partition) at `/home`. This file is set up automatically by the Linux installation process, but you may need to edit it if you add a hard disk or need to change some mount options.
- ✓ `/etc/group`—This file defines Linux *groups*—collections of users with similar access rights to the system. Even on a single-user computer, groups can be important because Linux uses them to track some of the programs it runs in the background.
- ✓ `/etc/inetd.conf`—This file controls `inetd`, which is one of two common *super servers* on Linux. (The other common super server is `xinetd`.) A super server launches other servers, the advantage being that a super server can be much smaller than the servers it launches, thus saving memory if a server isn't often being actively used. Super servers can also enhance security by providing preliminary checks on a connection—say, to weed out connections from known miscreants.

- ✓ `/etc/inittab`—As already noted, this file controls `init`, the first process.
- ✓ `/etc/modules.conf`—This file controls how Linux should load *modules*—that is, kernel drivers and other kernel code that's not part of the main kernel file. Some distributions, such as Debian and Gentoo, build this file automatically from files in a subdirectory, such as `/etc/modules.d`. On such distributions, you should edit only the files in the subdirectory rather than edit `/etc/modules.conf` directly.
- ✓ `/etc/passwd`—This file defines user accounts. As such, it's extremely important for normal functioning of the system and you shouldn't mess with it unnecessarily. Chapter 9 describes Linux account management.
- ✓ `/etc/printcap`—This file defines printer definitions for the old BSD LPD and LPRng printing systems. Some programs look into this file to determine what printers are available, so even CUPS maintains it.
- ✓ `/etc/shadow`—This file is a supplement to `/etc/passwd`; it holds encrypted passwords and supplementary account information that's not stored in `/etc/passwd`. This file is very sensitive, in a security sense. If a cracker gains access to `/etc/shadow`, the cracker may be able to acquire some user passwords. Thus, you should *never* send `/etc/shadow` over the network in an unencrypted form or copy it to a floppy disk and leave it lying around.
- ✓ `/etc/xinetd.conf`—This file controls `xinetd`, one of two common super servers on Linux. In most cases, actual configurations reside in the `/etc/xinetd.d` subdirectory; `/etc/xinetd.conf` just loads these server-specific configurations and sets some defaults.

This list is far from complete; it just hits upon some of the most important files and subdirectories in `/etc`. Many other obscure but important subsystems have configuration files in `/etc`. Also, note that the exact contents of `/etc` vary from one Linux system to another. In part, this is because different distributions provide different configurations. Even within a distribution, though, different selections for software to be installed results in different contents of `/etc`.

Ultimately, system degunking often involves editing files in `/etc`. Most of these files are plain-text files, so you can edit them with your favorite text editor. You shouldn't do so unless you know what you're doing, though, because incorrect changes can cause serious problems. (Various chapters in this book guide you through changes to files in `/etc`.)

TIP: After installing Linux, try backing up the entire `/etc` directory tree. You can copy the entire directory tree to the `/root` directory, which is the home directory for the

root user, by typing `cp -a /etc /root as root`. You might also want to create a backup on a Zip disk or other removable disk. Most modern Linux distributions /etc directories won't fit on a single floppy disk, though. Making a backup ensures that you'll be able to restore an original working file should an error creep in that causes the system to seriously misbehave.

Checking Program Configurations

Checking program configurations on a system-wide basis usually involves looking at files in /etc, or possibly in some other directory. The files you'll examine vary from one program to another, though, so you'll need to consult the program's documentation or third-party documentation (such as this book) for help. The configuration files, like user configuration files for programs, are often named after the programs themselves. User programs often look for user configuration files first and then use a global configuration file as a backup. Alternatively, they may mix the two in some way, such as loading global options and then using user-level options to modify these global options.

Earlier, in "Degunking Program Configurations," I said that it's sometimes helpful to delete a user configuration file. Doing so makes the program create a new file in its place when it's next run. This advice is *not* valid for most system configuration files. If a system configuration file is corrupt, you must fix the problem rather than delete the file and hope the program will create a valid one in its place. Fortunately, you can usually find sample files on the Internet, although they must often be tweaked for your system.

Linux Package Management

Most Linux distributions use a *package management* system. These systems enable you to install programs relatively easily, often by downloading a single file and then typing a command to install the package. The package management system keeps track of the individual files installed with the package, thus enabling you to easily uninstall the package at a later date. Package management systems also track *dependencies*—requirements of one package for other packages—and inform you of them if they're unmet. For instance, suppose you want to install SomeWriter 2.3 but it depends upon SomeLibrary 1.7, and SomeLibrary 1.7 isn't installed. When you try to install SomeWriter, the package management system will tell you about the unmet dependency so that you can track down SomeLibrary 1.7 and install it first.

Overall, using Linux package management tools is a great way to help avoid creating gunk. By tracking dependencies and installed files, package management tools make it easier to keep your system clean and keep all of the software

in sync. In practice, of course, package management has its downsides, but for the most part, these tools work well.

Not all Linux systems use the same package management system. Two are the most common:

- ✓ **RPM**—The RPM Package Manager (RPM) was created by Red Hat and has been adopted by Conectiva, Mandrake, SuSE, Yellow Dog, and various others. (Fedora, being a Red Hat variant, also uses RPMs.)
- ✓ **Debian packages**—The Debian distribution originated this package format, which is also used by various Debian derivatives, such as Lycoris and Xandros.

A few distributions use their own unique package management tools. Most notably, Slackware uses tar archives with special package-tracking tools and Gentoo uses its own system based on source code built on the target system itself.

The main command-line tool for dealing with RPM packages is called `rpm`. It provides options that enable you to install, uninstall, query, and otherwise manipulate package files and installed packages. The Debian equivalent is called `dpkg`, but Debian also provides various tools built atop `dpkg` in order to extend its functionality. Most important is the Advanced Package Tools (APT), which enables you to install a package and all its dependencies over the network and track updates to software. APT has also been ported to RPM-based systems, and most RPM-based distributions now provide APT or similar tools. Chapter 6 describes how to use these tools to clean out unnecessary programs and keep software up-to-date.

Degunking Hardware and Drivers

In many respects, gunky hardware and drivers can be the most frustrating types of gunk. Hardware problems often result in inconsistent behavior, which can make tracking down the problem very difficult. In Linux, having the *wrong* hardware is occasionally a problem—Linux driver availability, although quite good overall, isn't quite up to the level of driver availability for Windows, and some Linux drivers don't support all of a device's features.

Fixing Gunky Hardware

Several types of gunky hardware exist:

- ✓ **Inadequate hardware**—Hardware can work according to specifications but be inadequate for your purposes. For instance, your computer might not have enough RAM to run all the programs you want to run.

- ✓ **Flaky hardware**—One of the most frustrating problems is flaky hardware, which works fine one minute but fails the next. This can be the result of weak connections in cables or plug-in cards, broken traces on circuit boards, or other issues. Unfortunately, some types of driver bugs can create symptoms that look a lot like flaky hardware, which can complicate your degunking efforts.
- ✓ **Broken hardware**—Sometimes hardware is out-and-out broken. A printer might not respond at all to the computer, or a hard disk might not spin up. One of the trickier parts of dealing with such problems is verifying that the hardware is actually broken rather than disconnected from the computer or otherwise misconfigured.
- ✓ **Incompatible hardware**—Hardware can be incompatible with Linux because of missing or inadequate drivers. In truth, this category straddles the line between hardware problems and driver problems.

In many cases, the best approach to dealing with gunky hardware is to replace or upgrade it. Some types of hardware can be easily upgraded. For instance, you can add RAM to a computer with inadequate RAM, add a USB 2.0 card to a computer that has only USB 1.x ports, or add a new hard disk for extra space. Other problems can only be corrected by replacing the component, or even the entire computer. Chances are you won't be happy running any modern Linux distribution on a 386 computer, for instance, at least not as a desktop system or server. Such a computer should be ditched entirely, or at least shifted into a dedicated low-stress position. Buy a modern replacement computer and you'll be much happier. Some hardware problems can be corrected by replacing just one or two components, such as replacing an old CD-ROM drive with a recordable DVD drive or upgrading a scanner to a more modern unit.

One of the trickiest problems when dealing with hardware gunk is in tracking down the defective component. What do you do if you turn the computer on and it doesn't boot? Certain symptoms can be revealing, but they require a great deal of experience to interpret. For instance, where the computer goes astray during its failed boot process can tell you a lot about the problem—a bad CPU, bad RAM, bad hard disk, or something else. A thorough coverage of such issues would be a book unto itself, so I can't delve into all the details.

I can, though, suggest that you take a scientific approach to tracking down hardware problems. Create a hypothesis, devise a test of that hypothesis, implement the test, and note the results. (I recommend taking written notes as you do this—you can easily forget these things once you get deep into a debugging session.) For instance, suppose your system's not booting. You might hypothesize that the problem is bad RAM. You could then replace the RAM if you

have spare RAM, or perhaps swap out individual RAM modules if your computer has more than one, and observe the results. If removing or replacing the RAM fixes the problem, your hypothesis is faring pretty well, but if no amount of RAM adjustments helps, chances are something else is going wrong.

The Kernel and Other Drivers

Driver problems can be difficult to separate from hardware problems. To be sure, sometimes a problem is clearly caused by one or the other, but telling which is to blame can be tricky sometimes, particularly if you don't have spare components to test. Fortunately, Linux's drivers seldom cause outright failures or system crashes, although some drivers marked "experimental" are much riskier than others.

In dealing with driver gunk, you should first understand where drivers reside in Linux. Most Linux drivers are literally that—they're drivers that are part of the Linux kernel. These drivers "talk" directly to most of the hardware that's installed inside a computer's case, including hard disk controllers, RS-232 serial ports, parallel ports, USB ports, IEEE-1394 ports, the keyboard, various types of mouse ports, and network ports. These drivers can either be compiled directly into the main kernel file or be compiled as separate loadable kernel modules (controlled via `/etc/modules.conf`). In most cases, either approach works well. Most distributions arrive with most drivers compiled as modules because this helps keep the main kernel file's size manageable. When you recompile your kernel, though, you might build drivers you know you'll always need directly into the kernel file; this way, you don't have to worry about configuring them in `/etc/modules.conf`.

Some Linux drivers reside outside of the kernel. Most of these drivers are for devices that interface to the computer via a device that is supported by the kernel. For instance, a program called Ghostscript (<http://www.cs.wisc.edu/~ghost/>) provides printer drivers and the Scanner Access Now Easy (SANE, <http://www.sane-project.org>) package provides drivers for scanners. Printers and scanners both interface via ports that are managed via kernel drivers, such as USB or parallel ports. Perhaps the most notable class of non-kernel drivers, though, is X server video drivers. In most cases, X drivers are granted more direct access to the hardware than are other non-kernel drivers. Some X drivers, though, interface via *frame buffer* drivers. This is a mechanism that uses low-level video drivers in the kernel; these drivers provide a virtual display for X to use. Frame buffer video drivers for X tend to be slower than direct access, but on occasion they're more reliable. Frame buffer drivers are particularly popular outside of the x86 world because XFree86 and X.org-X11 have traditionally been weak in supporting non-x86 video hardware.

No matter where the driver exists, you can take several types of degunking actions:

- ✓ **Upgrading buggy drivers**—Buggy drivers can be very frustrating, in part because they can crash the system and in part because they can be very hard to tell apart from defective hardware. Fortunately, Linux drivers are seldom buggy, but when they are, you should look for an upgrade. In the case of kernel drivers, this usually means upgrading your kernel, but sometimes you can find precompiled updated driver modules.
- ✓ **Switching drivers**—A few devices are supported by more than one driver. This is particularly common in the arena of printer drivers. Sometimes switching from one driver to another will improve performance or reliability.
- ✓ **Changing driver configurations**—Sometimes you must pass special options to drivers to get them to work correctly. For kernel drivers, this is done via your boot loader or the `/etc/modules.conf` file, depending on how the driver was compiled. For printer, scanner, or X drivers, it's done via the relevant subsystem's configuration files. In any case, you'll need to research your device and the problem to find precisely how to pass any special options to the driver.
- ✓ **Making Linux filesystem changes**—Linux uses files in `/dev` to interface to many types of hardware. Sometimes you must create a file in this directory, or change permissions or ownership of an existing file, to enable ordinary users to use the hardware. For USB devices, making changes to files in `/proc/bus/usb` may be necessary instead.

Additional Sources of Information

Linux is a technically sophisticated and capable OS, but it's not quite as good in the documentation department. Although Linux does have some embedded help tools, they aren't as easy to use as are those in some OSs. Furthermore, many of these tools were designed for experienced Linux users and system administrators and therefore may not be of much help if you're really puzzled about how something works. For these reasons, knowing where to go for help is very important for Linux users. Several types of resources exist:

- ✓ **Linux man pages**—One of Linux's embedded help tools is its man page system. This system uses the `man` command (short for *manual*) to display information on a topic whose name you type after the command. For instance, to learn about `man` itself, type `man man`. The result is a page-by-page listing of information on the command, file, system call, or other feature you specify. Press the spacebar to page forward and press the Q key to exit. (The `man` system uses `less` to display information, so type `man less` to learn more about its paging commands.)

- ✓ **Linux info pages**—The info page system is the successor to man, at least in the eyes of some developers. The main advantage of info pages is that they use a hyperlinking technology to break information into small chunks that can be tied together, unlike the monolithic man pages. Type **info info** at a command prompt to learn more about this system.
- ✓ **The Linux Documentation Project**—A group of Linux users has created a set of semiformal Linux documentation, available online at **<http://tldp.org>**. Documents collected here include HOWTOs, which are short or medium-length task-oriented documents; Frequently Asked Questions (FAQs), which answer common questions; and Guides, which are longer (often book-length) documents on entire Linux subsystems, such as networking or security. The quality of this documentation is variable; some of it's excellent, but some of it is mediocre. Some of it was also very good in the past but is now obsolete.
- ✓ **Usenet newsgroups**—The Internet has long supported an online system of forums known as *Usenet*. This tool can be accessed using a *news reader* program, such as KNode (**<http://knode.sourceforge.net>**) or Pan (**<http://pan.rebelbase.com>**). You point the news reader to your Internet service provider's (ISP's) news server to access the newsgroups. (Consult your ISP for an appropriate address.) You can then read thousands of different newsgroups, many of which are devoted to Linux. Examples include **comp.os.linux.misc**, **comp.os.linux.setup**, and **comp.os.linux.networking**. You can usually get an answer to a question within a few hours on a newsgroup.
- ✓ **Mailing lists**—An Internet mailing list is similar to a Usenet newsgroup, but it uses a subscription-based model and e-mail rather than a news reader. When you subscribe to a mailing list, you'll begin receiving e-mail messages from the group, and you'll be able to post messages to the group. Many distributions and software development projects maintain mailing lists. Consult their Web pages for information on joining.
- ✓ **Web forums**—With everything moving toward the Web, it's no surprise that a Web-based variant on Usenet and mailing lists exists: Web forums. These discussion groups can be accessed with a Web browser. They're often run by distribution maintainers, although others, such as Yahoo!, also run Web forums. Some Web forums support delivery in mailing list form, linking the two into one unified discussion forum.
- ✓ **Web sites**—Linux Web sites are too numerous to mention. Good starting points include **<http://www.linux.com>** and **<http://www.linux.org>**. Doing a Web search on *Linux* and some topic can be a good way to find information.
- ✓ **Magazines**—Several Linux magazines exist. In the United States, *Linux Journal* (**<http://www.linuxjournal.com>**) and *Linux Magazine* (**<http://www.linuxmagazine.com>**)

www.linux-mag.com) are prominent examples, but *SysAdmin* (<http://www.samag.com>) is a more general magazine that nonetheless carries a great deal of Linux-relevant content.

- ✓ **Books**—You're holding one, so obviously you still value them. Books have long publication lead times, which can be a drawback, but they also often present the clearest and most in-depth coverage of particular topics.
- ✓ **User groups**—By computer terms, user groups are ancient. They involve people meeting *in real life* (imagine that!) to discuss their favorite computer topics. Linux user groups (LUGs) exist in many large cities. Check <http://www.linux.org/groups/> for a geographical listing. LUGs may not be very helpful for getting instant help, but they can be useful resources to learn about Linux generally.
- ✓ **People**—Friends, neighbors, and co-workers can all be invaluable resources when it comes to Linux—provided they know anything about the OS!

Overall, you're not alone when dealing with Linux gunk. Chances are others have been down the path you're traveling, and consulting these resources will help you clean up your configuration.

The Degunking 12-Step Program

Here is the basic 12-step degunking process that you'll follow in this book:

1. Delete files you don't need and better organize your remaining files (Chapter 3).
2. Select the desktop environment that best fits your needs and configure it properly to optimize the environment (Chapter 4).
3. Configure the settings for the common applications you use (Chapter 5).
4. Get rid of the unused packages on your system (Chapter 6).
5. Fine-tune the performance of the applications you use (Chapter 7).
6. Manage processes on your system, including dealing with misbehaving programs (Chapter 8).
7. Clean up the user accounts and system administration accounts on your system (Chapter 9).
8. Set up a system to reduce your e-mail spam and worms (Chapter 10).
9. Improve your network security (Chapter 10).
10. Fix any problems with hardware and drivers that you might have (Chapter 11).
11. Optimize your X configuration (Chapter 12).
12. Back up your system on a regular basis (Appendix A).

Summing Up: A Mindset for Degunking

You should begin your degunking process by preparing yourself mentally. This task actually begins with everyday computer use. You should strive to keep your configuration as simple and uncluttered as possible because unnecessary complexity can lead to problems. Safety is another important issue—one of the worst types of gunk is an intruder in your system, which can be like a pack of wildebeests trampling mud through your living room. Understanding how Linux is put together will help you when it's time to clean up. You'll be better able to locate appropriate user or global configuration files, track down hardware problems, and so on if you understand the basic shape of the OS.



Degunking User Files

Degunking Checklist:

- ✓ Determine and track how much disk space you're using.
- ✓ Identify and catalog the files that are on your system by using Linux commands and utilities.
- ✓ Use search tools to look for specific files on your system.
- ✓ Determine what types of files you want to keep and what files you want to delete or archive off your computer.
- ✓ Set up a good organization system for the files you want to keep and create the directories and subdirectories you need to store your files.
- ✓ Understand dot files and how they fit into the picture.

This chapter is the first of three that deals with user environment degunking. Because almost everything in Linux is a file, it's appropriate to begin with degunking files. For the most part, this means keeping your home directory organized and uncluttered. Typically, this directory is named after your account and located in the `/home` directory. For instance, if your username is `felix`, your home directory is likely to be `/home/felix`. This directory and all its subdirectories are your private domain, even if the computer has hundreds of other users. Keeping it tidy will help you do your work efficiently by making it easier to find your files and by reducing the odds of running out of disk space.

Track Your Disk Usage

The first step to maintaining a gunk-free set of user files is to know how your disk space is being allocated. If you've got 50GB for user files and 40GB of that is being consumed by a certain set of files, you might want to concentrate your efforts on the set of files that is taking up most of the space. You'll then be able to clear up disk space that you can use for other purposes. Linux provides various tools to help in this degunking task, ranging from programs that track overall disk space to programs that track disk use in particular directories. The most basic of these tools are text-mode, but GUI programs are also available.

Tracking Overall Disk Usage

One of Linux's quirks is that many of its basic text-mode commands are bizarre letter combinations. Frequently, they're abbreviations or words with their vowels removed. One such command that is very useful for tracking disk use is `df`. This command reports on overall free disk space as shown here:

```
$ df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sdb10	5859784	4750024	1109760	82%	/
/dev/sdb12	2086264	985872	1100392	48%	/opt
/dev/hda13	2541468	295804	2245664	12%	/usr/local
/dev/hda9	15361340	12775628	2585712	84%	/home
/dev/hda6	101089	19908	77006	21%	/boot
none	256528	0	256528	0%	/dev/shm
speaker:/home	6297248	4850716	1446532	78%	/speaker/home
//speaker/mp3s	17156608	8094720	9061888	48%	/speaker/smb/mp3s

Notice that the output of `df` provides information on all of the partitions, removable disks, and network mounts. Each column provides a particular piece of information:

- ✓ **Filesystem**—The first column shows the device filename or network identifier for the filesystem. This information most likely isn't very useful for tracking disk use, but it can be handy for some purposes, such as resizing a partition.
- ✓ **Filesystem size**—The second column reports the size of the filesystem. By default, this size is shown in kilobytes (or 1K-blocks, as the column header reads).
- ✓ **Used space**—The third column shows the total disk space used on the filesystem. Again, this is in kilobytes by default.
- ✓ **Available space**—The fourth column shows the available space on the partition. It's equal to the filesystem size minus the used space.
- ✓ **Use percentage**—The fifth column reports the percentage of available space that's in use. Because the preceding three columns typically have very long numbers, this is a very helpful column in tracking your overall disk use. As a general rule, if a partition's use percentage climbs above 80 percent, you should look into cleaning it up or adding disk space.
- ✓ **Mount point**—The final column identifies where the filesystem is mounted. This information is critical in tracking disk use. For user files, you might only be interested in the value on the `/home` filesystem. If `/home` isn't present, look at the root (`/`) filesystem instead—but be aware that this may hold both user files and system files.

The output of `df` typically includes several different types of filesystems. As noted, the most important for user files is whatever partition holds `/home`. This may be a separate partition, or `/home` could be stored directly on the root (`/`) filesystem. The preceding example also showed several other local filesystems (those whose filesystem names began with `/dev/sdb` or `/dev/hda`), the `/dev/shm` device (a pseudo-filesystem used by Linux for internal purposes—you can safely ignore it), and a couple of network file-sharing filesystems.

NOTE: Although I'm describing `df` in the chapter on degunking user files, this command is also very useful for system degunking. System administrators can track overall system file use with this tool. For instance, the preceding example shows that the root (`/`) filesystem is 82-percent full. Perhaps it's time to clean out some unused packages or go looking for stray files elsewhere on the system!

Overall, `df` is an excellent tool for getting the “big picture” of how your disk is used. It can tell you which of your partitions is close to being full and which are underutilized. It's also very handy when dealing with removable disks, which

tend to fill up quickly compared to disk partitions. For your first degunking task, I highly recommend that you run this command and look over the report that is provided. Based on what you find, you can use this information as your game plan to get rid of files that you don't need and better organize your disk.

Although `df` in its basic form is useful, you can use various options to this command to extract still more information or otherwise make it more helpful:

- ✓ **Specify units**—Pass the `-h` or `-H` option to the `df` command to have it display disk use in kilobytes, megabytes, or gigabytes, with the units specified. This can help tame the rather lengthy units used in several output columns. The `-h` option uses power-of-two (1024 bytes per kilobyte, aka binary kilobytes) units, while `-H` uses power-of-ten (1000 bytes per kilobyte) units.
- ✓ **Use megabytes**—The `-m` option tells `df` to display all sizes in binary megabytes, but the individual lines aren't numbered, unlike when using `-h` or `-H`.
- ✓ **Summarize inode usage**—Passing `-i` causes `df` to summarize *inode* use rather than disk space use. Inodes are disk structures that Linux uses when creating files. Some low-level filesystems have a limited number of inodes, so tracking inode use can be helpful if your filesystem has a large number of small files. Other filesystems don't have limited numbers of inodes, so `df` doesn't produce meaningful results in these cases (typically, values of 0 are reported in the output).
- ✓ **Display local filesystems only**—You might not be interested in network filesystems. If so, pass the `-l` (that's a lowercase *L*) to shorten the display by omitting network filesystems.
- ✓ **Filesystem type options**—You can limit the output to particular filesystem types (such as `ext2fs` or `ReiserFS`) by using the `-t fstype` option, where *fstype* is the filesystem type code (the same code used when mounting the filesystem with the `mount` command or as specified in the `/etc/fstab` file). The `-x fstype` option has the opposite effect; it omits the specified filesystem types. An uppercase `-T` option causes `df` to add a column telling you the filesystem type.
- ✓ **Specify a filesystem**—If you're interested in the usage of only a few filesystems, you can specify them on the command line, by listing either their device filenames or their mount points. For instance, `df /home /dev/sdb7` lists information on the `/home` filesystem and whatever filesystem is stored on `/dev/sdb7`.

Tracking Disk Usage in Specific Directories

Although `df` is a great tool for tracking overall disk use, another tool is needed to find out where space is being consumed *within* an individual partition. This tool is known as `du`, for disk usage. It's most often used by specifying options and one or more files or directories. Here's an example:

```
$ du --summarize reports budget
35240  reports
725    budget
```

The result is a list of the disk space consumed in blocks (each block is typically 1024 bytes). These values are the totals for all of the files or directories specified, including all subdirectories. For instance, if `reports` in the preceding example were a directory with several subdirectories, the reported disk use of 35,240KB for this directory includes the contents of all its subdirectories.

Obviously, `du` has great utility in tracking your disk use. By providing information on how much disk space is consumed by each subdirectory you specify, you can locate where you're using the most disk space, which in turn will help lead you to large files you may have forgotten, large caches of unnecessary backup files, or other sources of gunk you might be able to clean out.

The preceding example shows the use of an option (`--summarize`) with `du`. Several options can help you fine-tune `du`'s operation to best suit your needs:

- ✓ **Display file information**—Ordinarily, `du` doesn't report on individual files unless you specify them on the command line. You can pass the `-a` or `-all` option to have the utility report on file sizes. This tends to create very long output listings, and you can obtain the same information from `ls`.
- ✓ **Display summary**—The default behavior for `du` is to report on each subdirectory within a directory tree. For instance, if the directory `reports` has three subdirectories, and if you type `du reports`, you'll get back output with information on four directories: `reports` and each of its three subdirectories. The `-s` and `--summarize` options both trim this down so that the output includes information only on the directories you specify on the command line.
- ✓ **Display to specified depth**—Ordinarily, `du` follows all subdirectories however far they extend. The `--max-depth=n` option causes the program to stop after going *n* levels deep. The reported totals still include the contents of all the subdirectories, but the program doesn't report all the details. Using `--max-depth=0` is equivalent to `--summarize`.

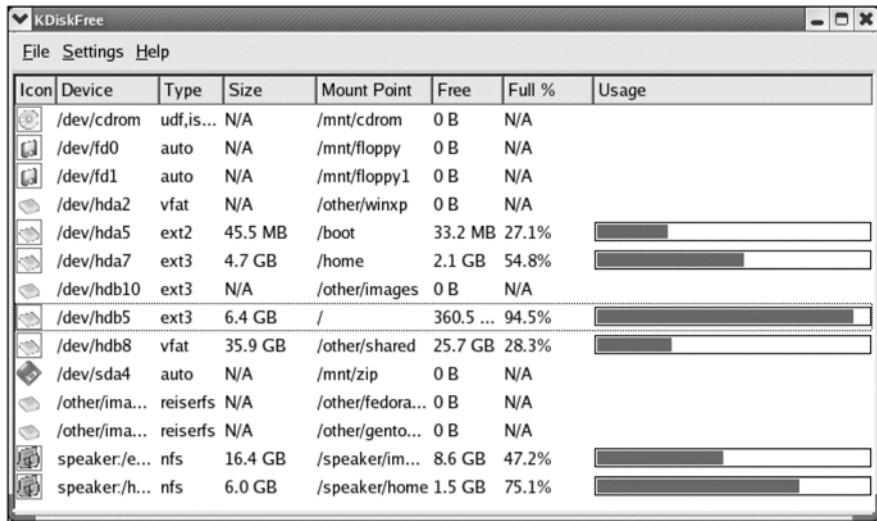
- ✓ **Omit subdirectories**—To obtain a total of disk space used by a directory *without* including its subdirectories, pass the `-S` or `--separate-dirs` option to `du`.
- ✓ **Display a total**—To sum up the space used on all the directories you specify, include the `-c` option to `du`.
- ✓ **Display units**—As shown earlier, `du` ordinarily doesn't specify units in its output. Passing the `-h` or `--human-readable` option causes `du` to append a power-of-two unit to the output. The `-H` and `--si` options are similar but display sizes in power-of-ten units.
- ✓ **Single filesystem**—If a directory you specify holds a subdirectory that serves as a mount point, `du` ordinarily includes all the files on the mounted filesystem. If you don't care about disk use on the mounted filesystem, though, you can pass the `-x` or `--one-file-system` option to `du`, which causes it to ignore mounted filesystems.

TIP: When you specify a wildcard (such as a question mark to match a single character or an asterisk to match any group of characters), Linux expands the wildcard and passes the results to the program. You can use this fact to get `du` to report on all the subdirectories in a directory: Type `du -s /*` to obtain a summary of the disk space used by all the subdirectories in the current directory. When typed in your home directory, this is an excellent way to discover where your personal disk space is being used.

Monitoring Your Disk Usage with a GUI Tool

Although `df` and `du` are great utilities for checking your disk usage from a Linux command prompt, you might prefer to take a more graphical approach. Fortunately, Linux offers tools to help you track disk use from a point-and-click interface. Some of these tools closely parallel `df` and `du` in capabilities, but others diverge more from their molds. Here are some of the more popular tools available:

- ✓ **KDiskFree**—This application, which is part of the K Desktop Environment (KDE), is essentially a GUI front end to `df`. It presents the same information as `df`, but in a GUI window with point-and-click controls to change the display. Figure 3-1 shows KDiskFree in action. The default KDiskFree display can be a bit cluttered because it shows partitions that aren't currently mounted (such as CD-ROMs and floppies that aren't presently in use) as well as partitions that are in use. KDiskFree can be found in the KDE menus on most distributions, or you can run it by typing `kdf` at a command prompt.
- ✓ **GDU**—The GNOME Disk Usage (GDU) program isn't officially part of the GNOME project, despite the name. You can find it at <http://>



Icon	Device	Type	Size	Mount Point	Free	Full %	Usage
	/dev/cdrom	udf, is...	N/A	/mnt/cdrom	0 B	N/A	
	/dev/fd0	auto	N/A	/mnt/floppy	0 B	N/A	
	/dev/fd1	auto	N/A	/mnt/floppy1	0 B	N/A	
	/dev/hda2	vfat	N/A	/other/winxp	0 B	N/A	
	/dev/hda5	ext2	45.5 MB	/boot	33.2 MB	27.1%	
	/dev/hda7	ext3	4.7 GB	/home	2.1 GB	54.8%	
	/dev/hdb10	ext3	N/A	/other/images	0 B	N/A	
	/dev/hdb5	ext3	6.4 GB	/	360.5 ...	94.5%	
	/dev/hdb8	vfat	35.9 GB	/other/shared	25.7 GB	28.3%	
	/dev/sda4	auto	N/A	/mnt/zip	0 B	N/A	
	/other/ima...	reiserfs	N/A	/other/fedora...	0 B	N/A	
	/other/ima...	reiserfs	N/A	/other/gento...	0 B	N/A	
	/speaker/e...	nfs	16.4 GB	/speaker/im...	8.6 GB	47.2%	
	/speaker/h...	nfs	6.0 GB	/speaker/home	1.5 GB	75.1%	

Figure 3-1

The KDiskFree program presents information on partition use in a GUI.

<http://www.ec-lyon.fr/~vincent/>. This program presents much of the same information as `du`. The indicator bars on the right of the display make it easy to spot the directories and files that consume the most disk space. Click their icons to expand the list to see which subdirectories or files are the culprits within the directories, then to locate where space is being used in the main directories' subdirectories, and so on.

- ✓ **KDirStat**—This program, headquartered at <http://kdirstat.sourceforge.net>, is another GUI `du` alternative, but it adds features that take better advantage of a GUI display, as shown in Figure 3-2. Specifically, it sorts files and directories by size and presents a graphical display of file sizes below the file list. Click on a square in this list to see what the file is. The program also offers various options under its Clean Up menu item. You can compress a file to save space, delete a file, and so on. You can extend this menu with user-defined options, so if you know of programs that can help save space for specific file types, you can call those programs directly from KDirStat.
- ✓ **XDiskUsage**—This program does some of the jobs of both `df` and `du`. When first launched, it presents a summary of disk space by partition, similar to what `df` presents. You can then click on a partition to see a summary of how its disk space is allocated. Click one of the directories presented in this window and you see how its disk space is tied up, and so on. Figure 3-3 shows the result. You can find this program at <http://xdiskusage.sourceforge.net>.

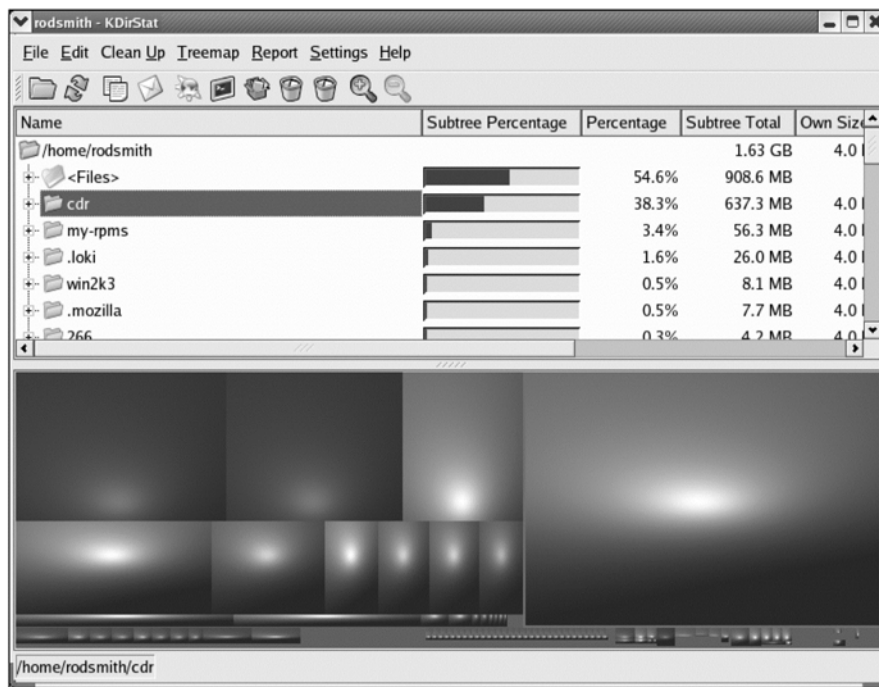


Figure 3-2

The KDirStat program presents information on file and directory sizes in a GUI.

Using one or more of these tools can be a great help in tracking down your gunk. KDirStat and XDiskUsage in particular are very handy. Their GUI displays make it easy to click on the largest files or directories, which can take you quickly to the files that are consuming the greatest amount of space. Of course, those files aren't always gunky—you might have a perfectly valid reason for keeping huge files. Nonetheless, they can be a good starting point, and large temporary files do sometimes manage to stick around without being purged. Examples include package files, CD-R image files, high-resolution scans, and music files. You might intend to keep these for a short time and then forget about them, but they chew up disk space quickly. These GUI disk space monitors will help you find them.

Identify, Delete, and Organize Your Files

Once you've located files that consume a lot of disk space, you may be faced with a few questions:

- ✓ What are the files?
- ✓ Are they really needed?

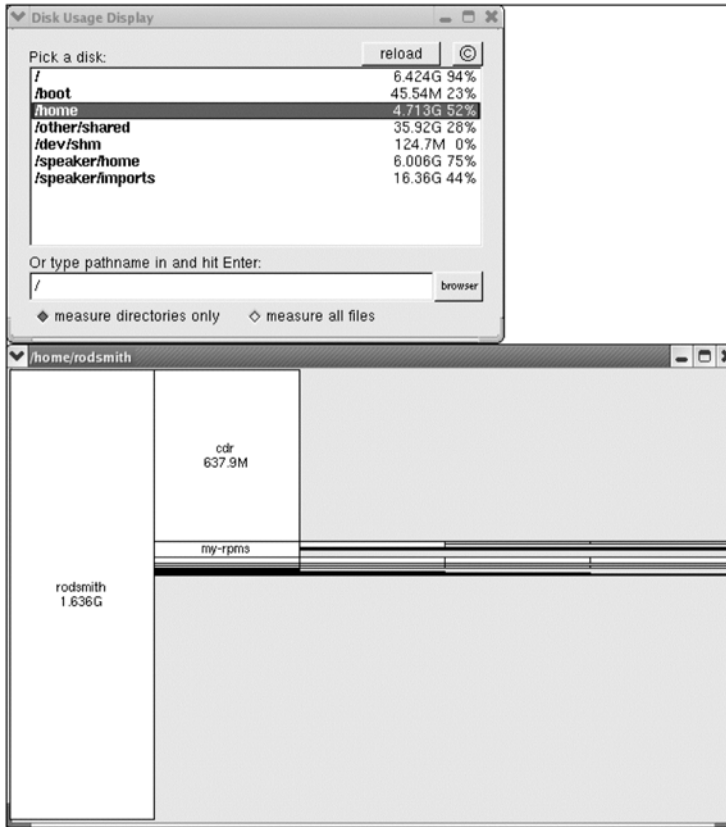


Figure 3-3

The XDiskUsage program handles tasks associated with both `du` and `df`.

- ✓ Can I delete them without causing any problems?
- ✓ If I delete a file that I end up needing, can I get it back somehow?
- ✓ If the file is one I want to keep, where is the best place to store it?

Unfortunately, the purpose of a particular file isn't always obvious. You might have created a file with a name that seemed obvious at the time but that eludes you now that you're a few years (or months or days) older and wiser. Sometimes, you won't even have created a file—it might have been created automatically by a program you ran. Several Linux tools can help you answer the question of what a file is. Other tools help you perform a sort of opposite task: locate files of particular types. Once you've located and identified a file, you must decide what to do with it: organize it (presumably better than it's organized now, lest you go through the process again) or delete it.

Identifying Files by Their Extensions

One of the simplest ways of identifying files is to examine their filename extensions. These are short (typically 1- to 4-characters) codes at the ends of filenames, preceded by a dot (.). For instance, in the filename `manual.sxw`, `.sxw` is the filename extension. Linux shares many filename extension standards with Windows and other operating systems (OSs), but some filename extensions are uncommon outside of the Linux and Unix worlds. The number of extensions in common use is huge, so I can't present a complete list here. Nonetheless, Table 3-1 summarizes some of the more common and important filename extensions you'll find on a Linux system, with particular emphasis given to Linux-specific extensions.

Table 3-1 Common Linux Filename Extensions

Extension	Explanation
<code>.bak</code>	Used by several programs to denote a backup file. Some programs append a tilde (~) to the ordinary filename, without changing the extension, to denote backup files.
<code>.bz2</code>	File compressed with <code>bzip2</code> .
<code>.c</code>	C source code file.
<code>.cc</code> , <code>.cpp</code> , or <code>.c++</code>	C++ source code file.
<code>.conf</code>	Configuration file. These are usually plain-text files with a special format and can be edited with text editors.
<code>.deb</code>	Debian package file; holds a complete program or other set of files. Described in more detail in Chapter 6, "Cleaning Out Unused Packages."
<code>.gif</code>	Graphics file in the Graphics Interchange Format (GIF).
<code>.gz</code>	File compressed with <code>gzip</code> .
<code>.h</code>	C or C++ <i>header</i> file. These files typically describe data structures for use by many other source code files.
<code>.htm</code> or <code>.html</code>	Hypertext Markup Language (HTML) document; a Web page.
<code>.jpg</code>	Graphics file in the Joint Photographic Experts Group (JPEG) format.
<code>.mp3</code>	Audio file in the Moving Picture Experts Group Layer 3 (MP3) format.
<code>.o</code>	Object code file (source code that's been compiled but not yet linked into a working program).
<code>.ogg</code>	Audio file in the Ogg Vorbis format. (This is an alternative to the MP3 format.)
<code>.pdf</code>	Portable Document Format (PDF) file; can contain text and graphics.
<code>.pl</code>	A Perl script.
<code>.png</code>	Graphics file in the Portable Network Graphics (PNG) format.

(continued)

Table 3-1 Common Linux Filename Extensions (Continued)

Extension	Explanation
.ps	PostScript file; used in Linux as an intermediary format for printing, among other things.
.py	A Python script.
.rpm	RPM Package Manager (RPM) file; holds a complete program or other set of files. Described in more detail in Chapter 6.
.sh	A shell script. (Shell script filenames often lack this extension, though.)
.so	A dynamic library file; used by regular program files to hold routines common to many programs.
.sxc	StarOffice/OpenOffice.org spreadsheet document file.
.sxw	StarOffice/OpenOffice.org word processing document file.
.tar	Archive file created by the <code>tar</code> utility; aka a <i>tarball</i> .
.tbz	Tarball compressed with <code>bzip2</code> .
.tcl	A Tool Command Language (Tcl) script.
.tgz	Tarball compressed with <code>gzip</code> .
.txt	Plain American Standard Code for Information Interchange (ASCII) text file.
.wav	Audio file.
.zip	Files archived and compressed with InfoZip, PK Zip, or a compatible tool.

GunkBuster's Notebook: Working with File Types

One problem with using Linux filename extensions is that they aren't a guarantee of the contents. You can easily rename, say, a PDF file to have an extension of `.zip`. The result can be confusing, of course, but such things can happen by accident or because somebody misunderstood what an extension meant.

Although not a filename extension, one particular filename deserves mention: `core`. Certain types of Linux program crashes automatically create a *core dump*, which is the contents of the memory the program had been using, placed in a file called `core`. (Core dumps are so called because they represent *core memory*, which is a very old term for what is today called RAM.) Core dumps are useful to programmers, who can use debugging tools to examine the program's state when it crashed to find clues to the cause of the crash. If you're not a programmer, though, core dumps are useless. Thus, if you find a file called `core` in your home directory tree, you can probably safely delete it. These files sometimes consume a lot of disk space, so searching for them with tools such as `find` (described later, in "Using `find`") can help you clear up a lot of wasted disk space.



NOTE: Unlike Windows, Linux doesn't use special extensions (such as `.exe` or `.com`) to identify executable program files. Certain scripts are conventionally identified by extensions, but this isn't required, and it's seldom done with binary programs. Instead, Linux identifies executable programs by a special file permission bit (the **execute bit**).

Identifying Files Types Using `file`

Assume you're cleaning house and you delete several files you know you don't need. You come across one file, though, named `sunflower.ras`, which is a thorn in your side. You've stared at the filename, pondered its meaning, and can't figure out what it is. You might even have loaded it into a text editor and you still can't decipher it. In the game of Chicken between you and the file, the file's winning.

Fortunately, a simple Linux program can come to your rescue. Appropriately, this program's name is `file`. Typically, you use this program by typing its name followed by the name of the file you want to identify, and the result is an identification:

```
$ file sunflower.ras
```

```
sunflower.ras: Sun raster image data, 1278 x 1024, 24-bit, no colormap
```

Unfortunately, the identification is sometimes a bit on the cryptic side, but it should provide some clues. In this case, several parts of the output indicate that the file is a graphics file—it's described as a "raster image," a size in pixels and bit depth are listed, and the lack of a color map is mentioned. In fact, Sun's raster image file format is one of dozens of graphics file formats. With this information in hand, you should be able to load the file into a graphics program such as the GIMP or XV and decide whether you want to keep it.

You can use the `file` command as I just described, but it also supports a number of options that modify how it works. Most of these are highly technical or relate to where the command looks for its own database of identifying file type features. Here I've listed the ones that are easy to use and can help you with your degunking chores:

- ✓ **Keep going**—Ordinarily, `file` stops when it finds a file type match. On rare occasion, though, `file` might not be able to distinguish between two or more file types. The result can be a misidentification of the file type. If this happens, try using the `-k` or `--keep-going` option, as in `file -k sunflower.ras`. This causes the `file` command to keep looking for matches and print all possible types for a file.

- ✓ **Look inside compressed files**—The `gzip` compression system is commonly used on Linux. Such files usually have `.gz` extensions. If a file you want to identify is so compressed, you can pass the `-z` or `--uncompress` option to identify the compressed file rather than identify it as a file that's been compressed with `gzip`.

The `file` utility is most helpful for identifying the type of a file—a graphics file, a word processing file, and so on. It won't help you read the data in the file, though. For that, you must load the file into an appropriate program. Worse, `file` is sometimes wrong in its identification. For instance, the StarOffice and OpenOffice.org programs create word processor files with `.sxw` extensions. These files are actually precompressed using the same file format as the popular Windows Zip archive utility, which throws off `file`:

```
$ file -k manual.sxw
```

```
manual.sxw: Zip archive data, at least v2.0 to extract
```

Note that even using the `-k` option didn't help in this case. The `-z` option wouldn't help, either; `gzip` uses a different compression format than the Zip format, despite the similarity in names. Despite this problem, `file` is a very useful utility, particularly when you're confronted with a moderately uncommon file type. It can often help you narrow down the options and at least point you toward a class of programs that might be able to read the file.

Finding Clues about Your Files Using strings

Sometimes, you just can't figure out what a file is all about by examining its extension or using `file` to determine the file type. In such cases, you can use the `strings` utility to do a little more detective work. This program scans a binary file and extracts ASCII strings from it. The `strings` program then displays these strings for your perusal. The result can often help you identify a file. For instance, you might notice text that looks like a letter to a friend, which would point to the file being a word processing document. Even with files that don't contain much plain text, using `strings` can be helpful. For instance, consider what happens when you use this command on a JPEG graphics file:

```
$ strings sunflower.jpg
```

```
JFIF
```

```
hSeptember, 2004
```

```
CREATOR: XV Version 3.10a Rev: 12/29/94 (PCD, PNG-1.2) Quality = 85,  
Smoothing = 0
```

Several lines of much less meaningful text follow these lines, but you can glean some useful information from these. The first line contains the string `JFIF`, which is a code used by JPEG files. The next lines contain comment text from the file, which in this case includes a date and an identification of the program that wrote the file (XV). Of course, you must understand these details for them to do you any good, but if you happen to have the knowledge, `strings` can help you identify a file type. (In this case, `file` will do the job as well, and with less knowledge on your part, but you might run across a file type that `file` can't identify but that you can, given a peek inside.)

Even if you know the file type, `strings` can be helpful. For instance, you might know that a file is a word processing file but be unable to load it for some reason—perhaps you don't currently have the right word processor installed on your system. You may be able to extract most or all of the file's text using `strings`, thus identifying the file's contents and enabling you to decide whether you want to keep the file or ditch it.

One problem with `strings` is that it tends to produce a lot of gibberish as output. This is because a large binary file is likely to have quite a few segments in it that look like ASCII text to a dumb program but that are useless to people. Thus, if you use `strings` and find yourself looking at a huge output, you may need to manage it in some way. One good way to do this is to pipe the result through the `less` pager:

```
$ strings sunflower.jpg | less
```

The vertical bar character (`|`) identifies a pipe, which links the output of the first program to the second program's input. In this case, the second program is `less`, which enables you to view a file's contents (in this case, the output of the `strings` command) one page at a time. Press the spacebar to move down a page. Various other keystrokes move back, search for text, and so on. Type `man less` to learn more about `less` from the Linux manual system.

Unfortunately, the `strings` command can't always recover useful text from files that contain text. The problem occurs when the text is encoded in some special way. For instance, the compression applied to OpenOffice.org files means that you won't be able to extract text from them with `strings`. The same is true for text files stored in compressed archives and encrypted files.

Other Methods of Identifying Files

In addition to these heavy-hitting or unusual methods of identifying files, several other tools and approaches are available to you. Some are useful only if

you've already got a good idea of the file's type but need to examine its contents to learn if it's a file you want to keep:

- ✓ **Doing less**—The `less` pager can be a good way to look at files. It can even display binary files, although the results are likely to be ugly. On the whole, though, `less` is best used to examine text files or those that are mostly text, such as word processor files.
- ✓ **Relying on a file browser**—Some file browsers can be configured to read part or all of a file to display a small image or summary of the file directly in the file browser or in a pop-up window when you move your mouse over the file's icon. This can be very effective for graphics files, which can then appear in small "thumbnail" form in the file manager's window. For text files, typically the first few lines of the file appear in a pop-up window.
- ✓ **Using an application**—Ideally, you should be able to load files into an application to view their contents. For instance, you might load a graphics file into a graphics viewer or editor. Sometimes you might not have the perfect application on hand, though, or it might be something of a heavy-weight and you don't want to twiddle your thumbs for a minute while it loads. In such cases, you may be able to get by with using a slimmer program that can load the target application's files. For instance, you might use `Abi Word` rather than `OpenOffice.org` or `Image Magick` rather than the `GIMP` to display a graphics file. File browsers often come configured with tools to load known file types into various programs; check your options and learn which programs launch most quickly.

Searching for Files

The degunking process not only involves identifying the files on your system but also locating some files in the first place. These might be files you want to delete or files that you intend to keep but that you want to relocate on your system to improve its organization. How will you know if you need to delete or move a certain file if you can't find it in the first place? As an example, you might know you've got a spreadsheet with old budget information somewhere, but you don't know where the file is. In such cases, you'll need to use a tool to locate your files. Two that are very useful are `find` and `grep`.

Using find

The program that's best for locating user data files from the command line is called, appropriately enough, `find`. To use this command, type its name, a starting path, and an expression. The expression can contain one or more keywords and values to search in various ways:

- ✓ **Depth limits**—The `-maxdepth levels` and `-mindepth levels` expressions limit searches to no more than, or no less than, the specified number of directory levels. This can be handy if you believe the searched-for file is not (or is) very deeply buried in your directory structure.
- ✓ **Searching mounted filesystems**—Ordinarily, `find` searches all subdirectories, whether or not they reside on their own mounted filesystems. Passing the `-mount` or `-xdev` option causes `find` to ignore directories that correspond to filesystems mounted within the search area.
- ✓ **Time searches**—The `-anewer file` option searches for files that were accessed more recently than the one you specify was modified, `-cnewer file` finds files whose status changed more recently than the one you specify was modified, and `-mnewer file` finds files that were modified more recently than the one you specified was modified. The `-atime days`, `-ctime days`, and `-mtime days` options search for files that were accessed, had their status changed, or were modified the specified number of days ago, respectively. A negative value for *days* finds files that are newer than the specified time, an explicitly positive value (such as +12) finds files that are older than the specified time, and values with neither positive nor negative markings (such as 12) finds files that are precisely the specified age.
- ✓ **Empty files**—The `-empty` option locates files that are empty (that is, that have a length of 0). This option also locates directories that hold no files. Both types of files are likely (but not certain) to be gunk.
- ✓ **Filename**—A common search is by filename. The `-name pattern` option is the most basic way to do this; it searches for a name that matches *pattern*. The *pattern* is a way to specify a pattern, similar to those used when typing filenames at a command prompt. Most simply, the *pattern* can be the exact filename you want to find. You can also incorporate wildcards, such as `?` to match any single character or `*` to match any string of characters. You may be required to enclose complex patterns in quotes. Variants on the `-name` search include `-iname`, which searches in a case-insensitive way; `-lname`, which searches symbolic links based on the name of the file to which the symbolic link points; and `-ilname`, which is like `-lname` but case insensitive.
- ✓ **Ownership**—The `-uid uid` and `-gid gid` options search for files owned by the specified numeric user IDs (UIDs) and group IDs (GIDs), respectively. The `-user username` and `-group groupname` options search for files owned by the specified user or group, identified by name rather than number.
- ✓ **Permissions**—The `-perm mode` option searches for files with the provided permissions (or *mode*), which can be specified in octal form (such as 644)

or in symbolic mode (such as `ug=rw`). Placing a plus sign (+) in front of the mode means that the search finds files for which *any* of the specified permission bits are present. A minus sign (-) means that *all* of the permission bits must be present. Using neither means that the file must have *exactly* the specified permissions.

- ✓ **File size**—The `-size n` option finds files that match the specified size, which is specified in 512-byte blocks by default, but you can append a `c` or `k` to search on bytes (characters) or kilobytes, respectively. If `n` is explicitly positive (such as `+500k`), `find` searches for files that are larger than the specified amount. If `n` is explicitly negative (such as `-250`), it searches for files that are smaller than the specified amount. When neither is the case (such as `300c`), only files that exactly match the specified size are found.
- ✓ **File type**—The `-type code` option searches for files that are of specific types. Among other values, `code` can be `c` or `b` for character or block device files, respectively; `d` for a directory; `f` for a regular file; or `l` (that is, a lowercase `L`) for a symbolic link.

That's quite the list—and it's not even complete! Clearly, `find` is a powerful command. Fortunately, just a few options can get you a long way. The `-name` and `-iname` options are particularly helpful. The various time and file size options can also be handy. For instance, suppose you want to find a file that you know you created less than a year ago and you're certain it was an OpenOffice.org word processor file with a filename that begins with `c`. The following command should find the file if you're at a command prompt in your home directory:

```
$ find ./ -iname "c*.sxw" -mtime -365
```

Using grep

Another tool for searching files is `grep`. Where `find` locates files based on surface features (the filename, file type codes, and so on), `grep` looks deeper: It examines the contents of files looking for patterns within the files' contents. To use it, type the command, any options you want to use, and a pattern to search for. You also usually include a file specification to search.

NOTE: When using the `find` command, you place the filename first after the command, but with `grep`, you put the filename at the end of the command. This difference can be a source of confusion. Personally, I'm constantly getting it mixed up!

Like `find`, `grep` supports many options; however, for basic file-finding duty, just a few are most important:

- ✓ **Ignore case**—Ordinarily, `grep` performs case-sensitive searches. The `-i` option tells the program to perform a case-insensitive search instead.
- ✓ **Produce a summary**—The usual `grep` output is a filename, followed by the line of text in which a match was found. (For matches in binary files, a message that the match was found appears.) If you just want filenames, though, you can pass the `-l` (that's a lowercase *L*) option to have `grep` omit the matching lines.
- ✓ **Search recursively**—Ordinarily, `grep` searches files that match the filename specification you provide. For instance, if you type `grep food ./*`, it searches for all files in the current directory that contain the string `food`. The `-r`, `-R`, or `--recursive` option (as in `grep -r food ./*`) performs a *recursive* search, in which all the subdirectories that match the file specification are searched. Such a search can be *very* time-consuming, though.

Using `grep`, you can find files based on their content. For instance, if you want to find a letter you wrote to your Aunt Mathilda, you might perform a recursive search for the string `Mathilda`:

```
$ grep -r Mathilda ~/letters
```

This command searches for all files in the `letters` subdirectory of your home directory in which the string `Mathilda` is present.

Unfortunately, `grep` isn't without its problems. As noted, recursive searches can take a long time because `grep` must read the entirety of every file it searches. (Some `grep` options that I've not mentioned can reduce this burden a bit, though.) This is particularly true when you search starting from a directory that's very large. The worst-case scenario is likely to be searching every file starting from the root (`/`) directory.

Like file-examination tools such as `less`, `grep` also suffers from an inability to perform useful searches on compressed, encrypted, or otherwise specially encoded files. If you wrote that letter to Aunt Mathilda in `OpenOffice.org`, for instance, `grep` won't be of much help because of `OpenOffice.org`'s use of compression in its default file format. Because `grep` searches textual data, it's unlikely to be of much use in searching files that contain basically nontextual data, such as graphics files or audio files. (A few exceptions do exist, though. For instance, you might search on textual comments included in nontextual files or on text fields in *some* types of graphics files.)

All in all, `grep` can be a very useful tool for searching certain file types—text files, textual configuration files, and so on. Its limitations aren't really a problem with `grep` so much as they are difficulties associated with searching certain file types.

Organizing Your Files

Now that you've learned how to locate and identify files, it's time to get them better organized. If you're staring at a home directory with lots of files and few or no subdirectories, or if you're finding your directory structure to be awkward, one way to approach the problem is to organize (or reorganize) your directory tree. Another approach, deleting files, is covered in the next section, "When to Delete Unused Files." These two approaches are complementary; you shouldn't try to use just one of them.

Organizing your files into subdirectories is easy, but it does require you to *create* subdirectories. From the command line, this can be done with the `mkdir` command, which creates (makes) a directory, hence the shortened name:

```
$ mkdir newdir
```

This example creates a directory called `newdir`. You can accomplish the same thing in a GUI file manager by selecting an option from the file manager's menu. Look for Edit > Create New > Folder or something similar. (The precise option name varies from one file manager to another.)

TIP: In Windows and Mac OS, filenames and directory names that contain spaces are common. Although Linux can handle such filenames, they're a bit awkward from the command line. You must either enclose the entire path in quotes or prefix each space with a backslash (\). For this reason, Linux users often use dashes (-) rather than spaces to de-limit words in multi-word filenames.

You should now be ready to create a sensible directory structure. Precisely what this will be depends on you and your own specific needs. Generally speaking, though, creating subdirectories for each project or file type can be a good starting point. For instance, you might create separate subdirectories for letters, financial documents, reports, and music files. Some or all of these subdirectories might have subdirectories of their own—say, business letters, personal correspondence, and letters to your legislators inside the directory you use for letters.

The idea behind this organization is to create a logical structure that makes it easy to find your files. Ideally, use the following tips as guidelines:

- ✓ Create a structure that results in a single logical location for any given document rather than several possible locations.
- ✓ Place a modest number of files in each directory—say, between half a dozen and three dozen. If your directories have too few files, your directory structure will end up being very deep, and accessing their contents will become tedious. If your directories have too many files, your directory structure will be conveniently shallow, but the files will be lost in the sea of files in each directory. Like Goldilocks, you want to find the comfortable middle ground.
- ✓ Use well-thought-out and descriptive directory names so that you can view a directory name and recall what you've placed in it. Avoid using a name such as `business-stuff`, and instead try a name such as `business-receipts`. Sometimes subdirectories can help with this; for instance, you might have a `business` directory with a `receipts` subdirectory.
- ✓ Try to arrange your directories so that they are easy to back up. For example, if you have a set of directories that store files that you are continually updating, you might want to arrange the directories so that they are grouped together. That way, you can easily back them all up on a regular basis.

With a directory tree constructed (or just partly constructed), you can begin moving files into place. If you are using a Linux GUI environment, you can do this using familiar drag-and-drop operations. If you're working from a command line, use the `mv` command, which doubles as a move and rename command. To move files, type the command name followed by the filename and the destination directory:

```
$ mv mathilda.sxw letters/personal-correspondence/
```

This command moves the `mathilda.sxw` file to the `letters/personal-correspondence` subdirectory. If you're starting from a disorganized clutter, you'll end up using this command or its GUI equivalents a lot!

TIP: *Linux text-mode shells usually provide a filename completion feature that can be very handy when manipulating files at the command line. Type a few characters (possibly as few as one) and then press the Tab key. Linux searches for all the filenames that might match and completes as much of the filename as it can. If just one filename matches the characters you've typed, Linux completes all the typing. This can save a lot of keystrokes when typing long filenames and directory names like*

personal-correspondence. *If there's only one file or directory in the letters subdirectory with a name that begins with a p, you can omit all the characters after the first one!*

When to Delete Unused Files

You'll likely need to shuffle files from one directory to another to help you keep your system organized and usable. Good organization can only help so much if your problem is that you're running out of disk space, though. What's more, even if you have plenty of disk space, unnecessary files will clutter your directory listings and perhaps require you to create more subdirectories (and perhaps a deeper directory structure) than you would otherwise need, thus making it harder to find the files that you do need. For all of these reasons, you should be prepared to throw out some old files, just as you'd chuck the moldy fruitcake that's been lurking at the back of your refrigerator since last December.

Deleting files can be difficult, particularly if you're not running short on disk space. It's easy to think of those files as potentially useful in the future, even if they aren't needed now. Maybe someday you'll write your memoirs and you'll need all those old letters that you've been saving. In most cases, though, this is wishful thinking; old files you've not touched in years are probably going to go untouched forever, so deleting them makes sense.

In the end, the decision to delete or not to delete a file is up to you. You might ask yourself some questions to help with this process, though:

- ✓ How long has it been since you've read or modified the file? The longer it's been, the better a candidate it is for the trash bin. A year might be a good deletion deadline, although this shouldn't be a hard-and-fast rule.
- ✓ Do you have a specific legal, business, personal, or other reason to keep the file? Tax records should certainly be kept around for longer than directions to a traveling circus, for instance.
- ✓ Can the data be reconstructed or obtained again? For instance, you can usually download a program file from the Internet at any time, so deleting program package files isn't a big risk. A photo from your digital camera, on the other hand, could be literally irreplaceable.
- ✓ How large is the file? Particularly when you're running low on disk space, a big file can be a real liability, but a small one isn't such a problem. (Lots of small files in a collection can add up to a big issue, though.)
- ✓ Is the file a backup file? Unless you're actively working on a project, an automatically created backup file is a good candidate for deletion.

These questions should help you evaluate each file you encounter and decide what to do with it: delete it or save it. Your evaluation could change over time, of course—something might be worth keeping today but not next month or next year. Thus, you may want to go through your files every now and then and clean them all up.

To delete files in Linux, you can either use the text-mode `rm` command or use drag-and-drop operations in most file managers. To use `rm`, type the command followed by the name or names of the files you want to delete, as in `rm letters/mathilda.sxw papers/quantum.doc`. Ordinarily, `rm` won't delete directories, though. To delete an entire directory tree, including all the files it contains, pass the `-r` option to `rm` along with the directory name, as in `rm -r olddir`. To delete just one empty directory, you can use `rmdir`, as in `rmdir olddir`. If you use a GUI file manager, you may need to select a menu option to “empty” the “trash can” before the files will be truly gone; simply moving the files to the trash puts them in a temporary holding area from which they can still be recovered.

CAUTION: By default, `rm` provides no prompts or warnings; it just goes and deletes everything you tell it to delete. The `rm` command also doesn't implement a “trash can” function; once you've deleted something with `rm`, it's gone! This means you can easily wipe out all the files in your home directory (or, if you run `rm` as root, all the files on the system!) by entering a typo. Some distributions configure `rm` to prompt you before deleting files, which tends to be safer. You can do the same by using the `-i` option to `rm`.

In some cases, a third option exists beyond keeping or saving a file: archiving it. You can save a file to a CD-R, Zip disk, or other removable medium. This practice gets the large file off of your hard disk and out of your main directory tree. If you make multiple copies or store the archives on good long-term archival media (such as CD-R discs), archiving data has the added advantage of keeping it safe—safer than it might be on hard disks, which do occasionally fail in devastating ways. You'll also be able to easily move the data from one computer to another, which can be handy in offices or when you replace your computer with a new one.

NOTE: Appendix A covers archiving files via backup software. Backing up your entire system is an important safety measure, and regularly backing up user files is also wise.

Clean Up Dot Files

Most Linux programs hide files and directories whose names begin with a dot (`.`)—so-called *dot files*—from view in file listings, file browsers, and file dialog

boxes. These files aren't inaccessible, though; they're just hidden from plain view. This is done as a way of keeping configuration files and other files that don't hold ordinary user data from cluttering file listings. Nonetheless, dot files can be a source of gunk. When you *do* need to modify or access dot files, locating the right one can be tricky because there are so many of them. On occasion, dot files or directories can also grow to abnormally large size, requiring deletion or other attention to fix.

What Are All Those Dot Files, Anyway?

Most Linux programs that enable you to set user-specified defaults create dot files, typically when they're first run. (Some programs require you to explicitly create their dot files, though.) Thus, as you use a system, chances are you'll accumulate dot files. Some of these will serve useful functions because they'll hold program defaults and other data that you really do need. Others will be useless clutter because you'll never run the program that created the file again.

Because they're hidden from view in most utilities, you may not be aware of how many dot files your system contains. Fortunately, many programs for displaying directories contain options enabling you to see dot files. For example, you can use the `-a` option with the text-mode `ls` command to view dot files:

```
$ ls -a
.          .config    .gnome      .ICEauthority  .rhn-
applet.conf
..         Desktop    .gnome2      .kde
.Xauthority
.bash_history .dmrc      .gnome2_private .metacity      .xemacs
.bash_logout .esd_auth  .gstreamer-0.8 .nautilus
.bash_profile .gconf     .gtkrc       .recently-used
.bashrc      .gconfd    .gtkrc-1.2-gnome2 .rhn-applet
```

NOTE: File browsers frequently have an option called *Show Hidden Files* or *Show Dot Files*, so look for such an option and enable it (or disable the option if it's called *Hide Dot Files* or something similar) to see dot files in a file browser.

As this example shows, a lot of dot files can exist in your home directory even if it is otherwise empty. This example has just one non-dot file (`Desktop`). Identifying the functions of these files is a critical first step in deciding what, if anything, to do with them. Table 3-2 presents some common dot files and their purposes to help with this task.

Table 3-2 Common Linux Dot Files

Dot File Name	Format	Purpose
<code>.bash_history</code>	File	History of commands typed at a Bash command prompt.
<code>.bash_logout</code>	File	Script that's run when logging out of a text-mode Bash shell.
<code>.bash_profile</code>	File	Script that's run when starting up a text-mode Bash shell.
<code>.bashrc</code>	File	Script that's run when starting up a text-mode Bash shell.
<code>.DCOPServer_*</code>	File	Filename varies with the computer's name; status file for the Desktop Communication Protocol (DCOP), which is used by KDE to help manage interprogram communications.
<code>.dirc</code>	File	KDE Display Manager (KDM) configuration file; specifies what desktop environment to run when logging in via KDM.
<code>.emacs.d</code>	Directory	Settings for Emacs, a popular text editor.
<code>.esd_auth</code>	File	Housekeeping file for the Enlightened Sound Daemon (ESD), which is part of Linux's sound subsystem.
<code>.fonts</code>	Directory	User-installed font files.
<code>.fonts.cache-1</code>	File	Listing of user-installed fonts in <code>.fonts</code> .
<code>.fonts.conf</code>	File	Xft font configuration file.
<code>.gconf</code>	Directory	Configuration files used by GConf, a centralized configuration tool associated with GNOME.
<code>.gconfd</code>	Directory	Status files used by GConf.
<code>.gimp-1.2</code> or <code>.gimp-2.0</code>	Directory	Configuration file, and perhaps temporary storage space, for the GIMP. Precise filename depends on the version of the GIMP being used.
<code>.gnome</code> , <code>.gnome2</code> , and <code>.gnome2_private</code>	Directory	Configurations for GNOME applications.
<code>.gtkrc</code>	File	Configuration information for the GIMP Toolkit (GTK+), which is a GUI development tool used by the GIMP, GNOME, and many other programs.
<code>.kde</code>	Directory	Configuration files for KDE applications.
<code>.macromedia</code>	Directory	Configuration files for Macromedia products, such as Flash Player.
<code>.mail</code> or <code>.Mail</code>	Directory	Mail messages; best managed with your mail program.
<code>.metacity</code>	Directory	Configuration files for the Metacity window manager, which is most commonly used by GNOME.
<code>.mozilla</code>	Directory	Configuration files and cache for the Mozilla and Mozilla Firefox Web browsers.
<code>.nautilus</code>	Directory	Configuration files for the Nautilus file browser used by GNOME.

(continued)

Table 3-2 Common Linux Dot Files (Continued)

Dot File Name	Format	Purpose
.rhn-applet	Directory	Housekeeping files for Red Hat's Update Agent program.
.rhn-applet.conf	File	Configuration file for Red Hat's Update Agent.
.signature	File	Traditional name for a <i>signature</i> file, which is a few (typically 1–4) lines of text appended to mail messages and newsgroup posts.
.ssh	Directory	Files associated with the Secure Shell (SSH) client program.
.themes	Directory	Themes for desktop environments.
.wine	Directory	Configuration and other files related to the WINE Is Not an Emulator (WINE) tool for running Windows programs in Linux.
.Xauthority	File	Automatically generated file that helps manage connections to the X server (Linux's GUI environment).
.Xclients and .Xclients-default	File	Script that's sometimes run when logging into a GUI session. (Not all GUI login methods use this script.)
.Xdefaults	File	Settings file that's used by certain X programs; can be used to set program defaults.
.xfce4	Directory	Configuration files for the XFce desktop environment.
.xinitrc	File	Script run when starting X from a text-mode login.
.xmms	Directory	Configuration file associated with the X Multimedia System (XMMS).

Table 3-2 is far from a complete listing of possible dot files, but it does give you an idea of what they can do. It also reveals a key fact: Most dot files' names are similar to the programs with which they're associated. Thus, you may be able to guess the name of the program that uses a dot file just by looking at the name of the dot file. Some names append `rc` or `.conf` to the end of the configuration name, so try omitting this if you see it.

NOTE: Two dot files (actually directories) deserve special attention. The `.` directory refers to the current directory, while `..` refers to the current directory's parent directory. This is why the text-mode command `cd ..` brings you to the parent of the current directory—you're moving into a subdirectory that's really the parent of the current directory.

Dot files and directories can become gunk in three main ways:

- ✓ **They become unnecessary.** If you run a program once, decide you don't like it, and then never run it again, its associated dot file becomes unnecessary clutter. Although the space used is likely to be small, the dot file will make it harder to find other dot files when you go looking for them.

- ✓ **They become too big.** This problem is most likely to affect dot directories, which can sometimes collect stray files. This can happen accidentally (say, if you mistakenly copy a big CD-R image file to a dot directory) or because you make changes that you forget to undo (say, adding personal fonts that you don't need but then forget about). Dot directories that are most likely to collect such detritus include `.fonts`, the `.gimp-*` directories, `.mail` or `.Mail`, `.mozilla`, `.themes`, and `.wine`. In most cases, the best solution is to clean the directories out using their associated applications. For instance, you might clean up your mail folders in your mail program or remove fonts using the KDE font manager. The GIMP dot directories and dot directories for Web browsers can collect temporary files. These programs often have options that enable you to limit the size or number of temporary files they collect. The `.wine` directory holds a stand-in for the Windows Registry, which can become quite cluttered if you run lots of Windows programs using WINE.
- ✓ **They hold bad configurations.** If you make poor changes to your program configurations, the dot files can become gunk because they're the receptacles for those configurations rather than because they're too big or unnecessary. The usual solution is to change the configuration in the programs in question. In extreme cases, though, you might want to delete the dot files. The programs will then create new dot files when you start them again. This approach is most helpful when a program won't start at all because its configuration file is in such bad shape. It has the drawback that you'll lose any customizations you've created.

Should You Delete Dot Files?

As a general rule, you shouldn't go around deleting dot files without cause. They may clutter up your home directory to some extent, but many of them are necessary. Mucking with them can cause problems, so you shouldn't do it unless you must.

Of course, if a dot file has grown to a huge size, you might want to investigate it. Look back at Figure 3-2. The list of directories in the top pane of that window is ordered according to directory size, and two of the directories visible in that list as top disk space consumers are dot directories: `.loki` and `.mozilla`. The `.loki` directory isn't listed in Table 3-2. It's associated with a now-defunct company that ported games to Linux, and it holds 637.3MB of files in saved games. Deleting some of those saved games will save a lot of disk space. The `.mozilla` dot directory holds a comparatively modest 7.7MB of data, mostly in cached images from Web pages. Deleting some of those images might save

some space, but it could also slow subsequent visits to the cached Web pages. Mozilla manages its cache automatically, so those files will be deleted in time—and replaced with other cached images. As noted earlier, though, you can change Mozilla's configuration to devote more or less disk space to its caches.

If a dot file is no longer necessary because you no longer run the program in question, of course it's a good candidate for deletion. If your system produced Figure 3-2, for instance, and if you stopped playing Loki games, you could safely delete the `.loki` directory entirely, not just clean it up a bit.

Unfortunately, I can't present a hard-and-fast rule about when to delete dot files. You'll need to make that judgment for yourself, based on your knowledge of the purpose of the dot files and your own need for disk space and desire to reduce clutter in dot files.

Methods of (Relatively) Safely Deleting Dot Files

If you're not sure of the function of a dot file, you might want to take some precautions before deleting it. The best approach is to rename or move the dot file rather than delete it. For instance, you might create a subdirectory (say, `dot-files`) and then use `mv` or a GUI file browser's drag-and-drop operation to move the dot files to this directory. You can then log out and log back in to check the effects. Of course, if you don't know what program used the dot file, you might not see the effects immediately; you might notice only after several minutes, hours, days, or even longer that you've lost the defaults in some program or that something else is different in your environment. You can then check the dot files—compare the dot files in your home directory proper against the ones you've moved into `dot-files` to see if any have reappeared. You might then consider reversing the process.

TIP: You might want to create backups of your dot files. Create a directory, as just described, and then copy (but don't move) your dot files to it. This can be helpful in the event that a dot file becomes corrupt or you change a configuration option and have trouble getting the original configuration back—you can copy the original dot file over the corrupted one. Be mindful of the disk space used by your dot file backups, though. You might want to delete some extraneous files (like saved games and cache files, if you can identify them) or even omit particularly large dot files if they aren't very important to you.

Generally speaking, it's best to delete dot files when you're not using a GUI environment or running any more programs than you must. That way, you

won't confuse any program by removing its dot file while it's running. At worst, the program will see that the file is missing when it's restarted; most user programs will then fall back on system defaults or create a new dot file.

Of course, if you've positively identified a dot file as belonging to a program that you don't run, these precautions may be excessive. Ultimately, you'll have to be the judge of that.

Summing Up: Cleaning Your Home Directory

You now should have completed four important degunking tasks for better managing your user files: tracking how much disk space you are using, identifying the files you want to keep, creating directories and subdirectories for storing the files you keep, and deleting files you don't want to keep. You also learned how to work with Linux dot files and how these files can become a source of gunk if you are not careful. Dot files deserve special consideration because they don't show up in most program listings and so they're easy to forget. Because they're program configuration files, they can require special treatment when you delete or modify them.

Linux provides quite a few tools, both text based and GUI, for managing files, locating files, creating and removing directories, and deleting files. You can even track your disk usage to help locate gunk that's consuming too much disk space and search for specific files and types of files to help localize specific gunky files.



Degunking Your Desktop Environment

Degunking Checklist:

- ✓ Know the features of your preferred desktop environment (GNOME, KDE, or something else).
- ✓ Use the best desktop environment features while avoiding those that simply suck the life out of the system by consuming CPU time or other resources.
- ✓ Keep an uncluttered desktop environment.
- ✓ Investigate the possibility of using an alternative to the popular GNOME and KDE desktop environments.

When you use Linux as a desktop OS, chances are you'll use Linux in its GUI mode. Linux's GUI consists of the *X Window System* (or *X* for short), but *X* alone is very spare. At a minimum, you'll use a *window manager* along with *X*. A window manager provides window decorations that also provide controls that enable you to drag windows around the screen, resize windows, and so on. Window managers also usually provide some way to launch programs. Window managers alone are fairly simple tools, though. Most people use entire *desktop environments*, which consist of window managers along with a selection of small *applets*—small programs that provide simple but useful features such as calculators and tools to change the keyboard repeat rate. The two most popular Linux desktop environments are the GNU Network Object Model Environment (GNOME, <http://www.gnome.org>) and the K Desktop Environment (KDE, <http://www.kde.org>). Both are capable desktop environments, but they've also both become quite complex, which means they're good gunk collectors. Knowing how to deal with this gunk, or bypass it entirely by using a simpler desktop environment or even a “bare” window manager, will help you keep your Linux working environment clean. In this chapter I'll show you how to tweak both GNOME and KDE so that you can make either of these desktops work better for you.

Tweak GNOME for Better Performance

As its name implies, GNOME was originally designed as a network-centric desktop environment. As it's developed, though, much of the original network-centric vision for the environment hasn't materialized, so GNOME and KDE have become remarkably similar in overall features. Knowing a bit about what you can tweak in GNOME will help you to improve its overall performance. You should also take steps to keep your GNOME desktop uncluttered. Knowing what you can and cannot safely change will help you to do this.

A Tour of Adjustable GNOME Features

I'll start by showing you some of the basic customization features that GNOME offers. This will help you be more productive with your desktop environment. Consider Figure 4-1, which shows the default GNOME 2.8.1 configuration on a Fedora 3 system. Although Fedora changes many GNOME defaults, they're not so strange as to be unrecognizable. (Previous versions of Fedora made more extensive changes to the standard GNOME configuration.)

Most configurable GNOME features can be set through the GNOME Preferences window, as shown in Figure 4-2. You can open this window by double-clicking the Start Here icon on your desktop and then double-clicking the

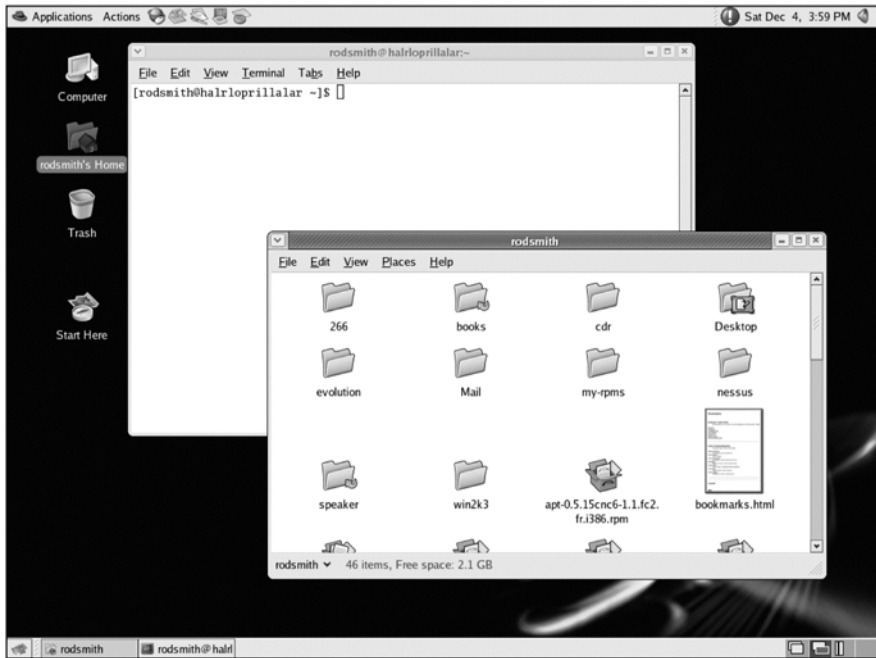


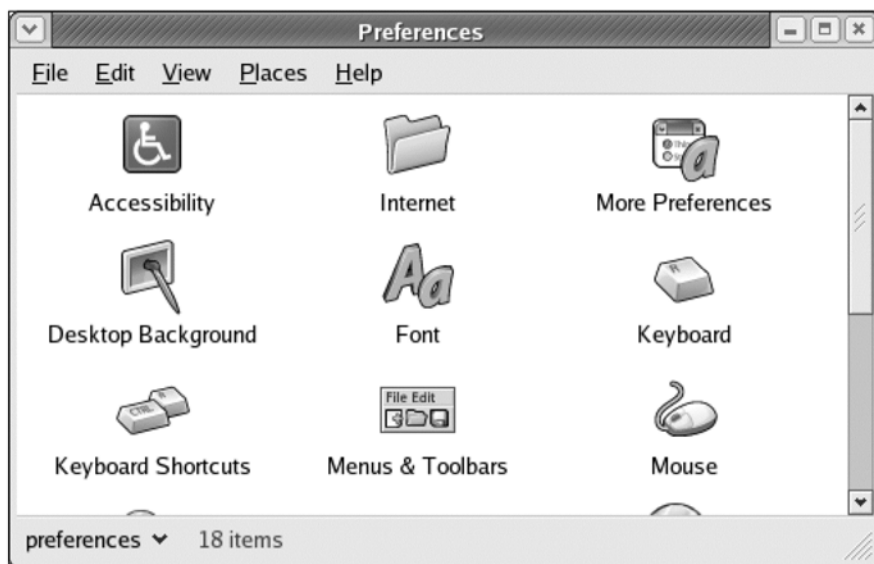
Figure 4-1

GNOME provides many features that can be customized to suit your preferences.

Preferences icon (this is called Desktop Preferences on some systems). You can access the individual items in this menu from the Applications > Preferences menu at the top of the default GNOME screen as well.

Features that you might want to degunk include the following:

- ✓ **Desktop Background**—Most distributions' default desktop backgrounds use a graphic image of some sort. You can change this image to any of several that are provided or point to your own image by starting Desktop Background and clicking the Add Wallpaper button. On low-memory systems, background images can consume resources you might prefer to devote to applications, so using a solid color may be a better choice. You can activate this choice and then select your color from the color picker by clicking the No Wallpaper option.
- ✓ **Font**—You can set several font-related options, including the fonts used in window titles and the default font in GTK+ applications. You can also set font rendering options, including whether to use *font smoothing*, aka *anti-aliasing*. These terms refer to the use of gray pixels near sharp turns in character shapes, which reduces the appearance of jagged edges. Fonts are described in more detail in Chapter 12, "Optimizing Your X Configuration." Some of these font options, such as the font smoothing setting, affect many

**Figure 4-2**

Various GNOME defaults can be set using small utilities.

applications. Others, such as the default application font, affect only GTK+ applications, such as those that make up GNOME.

- ✓ **Keyboard**—The main option of interest in this item is the ability to set the keyboard repeat rate. For some inexplicable reason, the repeat rate is often maxed out by default, which means that when you hold down a key, your screen will be filled with the character you're pressing. Reducing the keyboard repeat rate to a sane value by adjusting the slider can make the keyboard repeat feature a feature rather than annoying gunk. You'll need to experiment to find a value you like.
- ✓ **Mouse**—You can set the mouse tracking speed, configure the mouse for left-handed use (which reverses the roles of the left and right buttons), and adjust the double-click sensitivity from this item.
- ✓ **Password**—This item is very important because it enables you to change your account's password. If you leave your password unchanged for too long, miscreants might discover it and use it. Changing the password periodically will help keep these unwanted individuals out, or at least minimize the damage they can do by limiting the time they can spend using your account.
- ✓ **Removable Storage**—GNOME tries to make your life easier by automatically mounting removable media (floppy disks, CD-ROMs, and so on) when you insert them and opening a file browser on these disks. Some people find this behavior quite annoying, so it's configurable. You can tell the system whether to mount media, what to do when audio CDs are detected, and so on.

- ✓ **Screen Resolution**—Recent X servers support a protocol that enables users to change the screen resolution on the fly. If your display is too big or too small for your taste, you may want to adjust it using this tool. On the other hand, properly configuring X to start in the best resolution before any user logs in may be a better solution. This can be done using the X configuration file or GUI tools in the system administration area, as described in Chapter 12.
- ✓ **Screensaver**—GNOME provides a screen saver that displays dynamic patterns and, optionally, turns the display into a low-power mode after a period of inactivity. You can configure the patterns you like, set timing options, and tell the screen saver whether or not to use the power-saving mode with the Screensaver item.
- ✓ **Sound**—You can tell GNOME to start a *sound server*, which is a program that helps manage sound from multiple programs, and to enable system sounds. If sound doesn't work at all, though, you'll need to degunk sound on a system level. Unfortunately, this task can be a tricky one; Linux sound is still harder to configure than most other types of hardware.
- ✓ **Theme**—You can set several configuration options in one fell swoop from this item. It mainly affects the window border style, the desktop background, and the color scheme used by GTK+, but some themes include hints for fonts and other features. Theme selections are very personal—what one person considers just right could be another person's unbearably gunky theme.

Some of these tools accept files you can download from the Internet. The most notable is the Themes tool. Check <http://themes.freshmeat.net> for a collection of themes. For use in GNOME, investigate the GTK+ themes. Keep in mind that themes can be a source of gunk—if you install dozens of themes on your system, they'll consume disk space. I've even seen themes that make desktop environments crash!

GNOME supports configuration options in addition to the options available in the Preferences area. Most notable are the Panels—the bars at the top and bottom of Figure 4-1. You can add Panels, delete Panels, and modify the contents of Panels. By default, the top Panel in GNOME 2.8.1 includes a pair of menus (Applications and Actions) that enable you to launch programs and perform other tasks, a series of icons for launching specific programs, and a few tool tips that provide status and enable you to adjust features by clicking them. The bottom Panel typically includes a window-hiding button, a series of buttons corresponding to open windows, and a *pager*, which enables you to switch between virtual desktops.

To modify the Panels, right-click on them. This action produces a context menu with various options, but the precise options available depend on where you click. If you click in a blank part of the Panel, you'll be able to add items to the Panel, delete the Panel, or modify the Panel's properties (its size and location, for instance). If you click on an existing Panel icon, you'll be able to adjust the icon, rather than the Panel as a whole.

Tweaking File Browsing in GNOME

GNOME, like all major Linux desktop environments, provides a *file browser*—a program that displays files and directories in a graphical manner and enables you to copy, delete, rename, and otherwise manipulate files. GNOME's file browser is called Nautilus, and Figure 4-1 shows one Nautilus window open on the GNOME desktop.

Nautilus provides the sorts of features that users expect in a modern OS's file browser. Beyond simple file manipulation, Nautilus can provide previews of files' contents, display different icons for different types of files, link files to applications so you can load a file directly into the application by double-clicking its icon, and so on. Many of Nautilus's features are configurable. You can change some on a window-by-window basis from the View menu, but to change the defaults, you must select Edit > Preferences. This action brings up the File Management Preferences dialog box shown in Figure 4-3, in which you can set various defaults:

- ✓ **File display**—The Default View item on the Views tab controls how Nautilus displays files—as icons or in a list view. The latter includes extra information, such as file types, file modification dates, and file sizes. You can control precisely what information you want to display from the List Columns tab.
- ✓ **Arrangement**—The Arrange Items item on the Views tab controls how Nautilus orders files. You can sort by name, by date, by file type, and so on.
- ✓ **Icon sizes**—You can set icon sizes separately for icon and list views. The icon view also gives you the option of compacting the layout, which reduces the spacing between icons. Cramming more icons into a window can be handy if you've got a small display or if you like taking in as many files as possible in crowded directories, but it can lead to squashed-up and difficult-to-read icon lists if you go overboard.
- ✓ **Click behaviors**—The Behavior tab provides options relating to click and double-click behavior. You can configure Nautilus to open files when they're single- or double-clicked and tell the browser whether to open or run executable text files (such as scripts) when they're clicked.

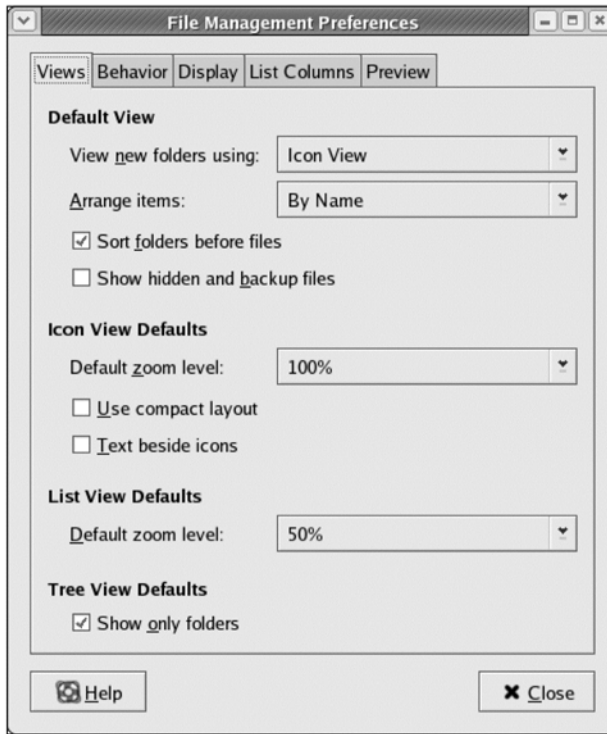


Figure 4-3

The Nautilus File Management Preferences dialog box controls the Nautilus defaults.

- ✓ **Browser windows**—The default Nautilus window is a simple display of file icons. Some people prefer a more elaborate display similar to that used by Web browsers, as shown in Figure 4-4. To use such icons, check the Always Open in Browser Windows item on the Behavior tab.
- ✓ **Trash options**—Nautilus maintains a “trash can” (the icon is on the desktop and is visible in Figure 4-1). The Behavior tab provides options that affect its operation—specifically, whether to confirm requests to delete the trash and whether to include a delete option that bypasses the trash.
- ✓ **Preview options**—The Preview tab provides options related to previews of files. For instance, in Figures 4-1 and 4-4, the `bookmarks.html` file is pre-viewed, meaning that a small version of the file is displayed in the file browser itself. Such previews can be handy, but they can also be very time-consuming, particularly if a large directory is filled with files that require a lot of time to preview, such as a directory with dozens of large digital photographs.

**Figure 4-4**

You can change the Nautilus display to include a Location field, Forward and Back buttons, and other Web browser-like features.

Careful selection of your file browsing options can greatly improve GNOME's overall performance. Pay particular attention to the preview options, which can suck your system's performance like a vampire if they're set incorrectly. In particular, be sure you're previewing local files only, and set a file size limit (the Only for Files Smaller Than option) that works well for your folders. You may need to experiment with these options to learn what works well. If your hard disk is particularly slow or if your directories contain lots of files, you may want to completely disable file previews.

Most other file browsing options are matters of personal preference. For instance, some people love browser-style windows, but others prefer the simpler traditional style. You'll simply have to try these options yourself to see which you prefer.

GunkBuster's Notebook: Setting Backgrounds and Emblems

Another feature of GNOME that can be tempting but can lead to problems if overused is the ability to attach emblems and backgrounds to icons and windows. Select Edit > Backgrounds and Emblems to see the Backgrounds and Emblems window shown in Figure 4-5. You can select patterns, colors, or emblems by

clicking the appropriate icon on the left of this window. The large right pane then shows a selection of styles for the item you've selected. Patterns and colors can be applied to directory backgrounds, while emblems can be applied to individual directory and file icons. In both cases, you drag the object you want into the window or over the icon.

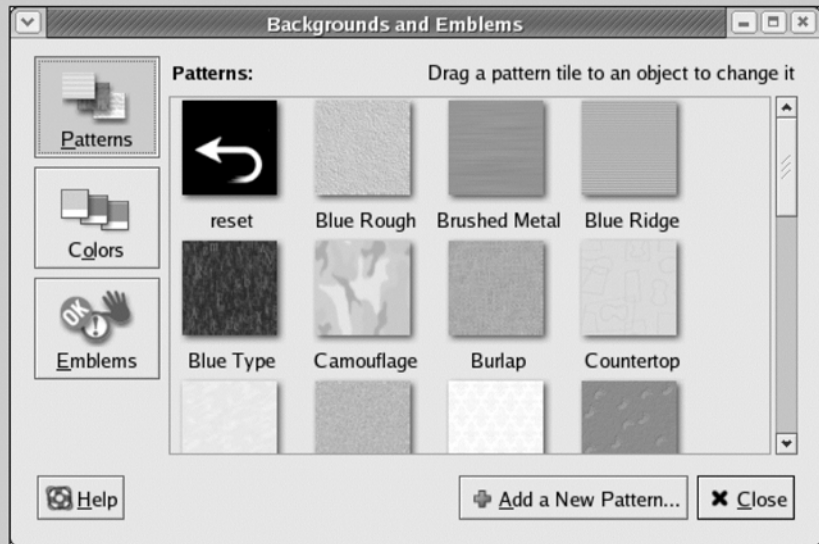


Figure 4-5

Backgrounds and emblems enable you to customize individual windows and file icons.

Backgrounds and emblems can be useful ways to help make certain files and directories stand out. For instance, ordinarily directories all have the same folder icon, which makes it hard to locate specific directories. You might attach emblems to important directories to make it easier to locate them in a window, which can speed up navigation of your system. Using patterns and emblems, in particular, requires a bit of extra disk access and so can slow down system performance, although probably not perceptibly in most cases. If you find you need to use a lot of emblems and backgrounds, this could be a sign that your files aren't well organized—perhaps you've got too many files in each directory. You may want to reconsider your filing strategy, as described in Chapter 3, "Degunking User Files."

Keeping Your GNOME Desktop Uncluttered

The GNOME desktop is actually maintained by Nautilus. The GNOME file manager is responsible for handling the icons you see on your desktop. Most distributions ship with a fairly simple default GNOME desktop, which contains icons that link to your home directory, the Start Here icon, the trash, and the Computer icon (which in turn provides access to removable media). You can add icons to the desktop, though. These icons might be shortcuts to specific subdirectories in your home directory, shortcuts to system or network directories holding common files such as music files, or tools for launching your favorite programs.

NOTE: *The GNOME desktop corresponds to the Desktop directory in your home directory.*

To create an icon, right-click on the desktop and select an appropriate option (such as Create Folder, Create Document, or Create Launcher) to place the selected type of icon on the desktop. You can also drag and drop an icon from a Nautilus window onto the desktop to move it there.

The trouble with all this activity is that it tends to create a cluttered working environment. As with a lot of degunking tasks, the challenge is in finding the right balance. A few program-launch and document icons on the desktop can help you be productive by enabling you to more quickly launch programs and access files. Too many, though, and you'll have a hard time finding the icons you want. You'll have to decide on an appropriate balance yourself, but as a general rule of thumb, try to keep the number down to a dozen or so.

TIP: *If you want quick access to a large number of programs or files, you might be able to store these links in folders on your desktop. For instance, 5 folders with 10 items each gives you reasonably quick access to 50 programs or files while keeping your desktop relatively uncluttered.*

Starting with a Fresh GNOME Configuration

Sometimes a GNOME configuration becomes badly corrupted. This is particularly common if you've switched distributions—one distribution might store files in one directory, whereas another stores them somewhere else. The result can be missing icons, programs that no longer run, and so on. This is a serious gunk problem, but it's relatively simple to fix, with an important caveat: You'll end up starting from scratch in your GNOME configuration. The idea is to delete your existing GNOME configuration files. Once you've done this, the next time you start GNOME, it will create new configuration files starting with the defaults,

which should be correct (or at least useable) for your system. Chapter 3 describes this approach in general, and Table 3-2 presents the names of common Linux *dot files* (hidden configuration files) that you can delete to reset various programs to their default configurations, including GNOME dot files.

Unfortunately, GNOME doesn't put its configuration files in one convenient location. As Table 3-2 reveals, you must delete or move several different dot files and directories to fully reset GNOME to its defaults: `.gconf`, `.gconfd`, `.gnome`, `.gnome2`, `.gnome2_private`, `.gtkrc`, `.metacity`, `.nautilus`, and `.themes`. You might not need to delete all of these files and directories, though. In particular, `.gconf` and `.gconfd` are related to configuration tools, `.gtkrc` is associated with GTK+, `.metacity` controls the Metacity window manager that GNOME uses, `.nautilus` stores settings for the Nautilus file manager, and `.themes` stores themes. If some of these components aren't giving you problems, you might try leaving their configurations in place. If you do so but still have problems, try deleting more configuration files until you find the files that are the source of the problem.

TIP: As described in Chapter 3, instead of deleting configuration files and directories outright, try moving them to a temporary location. That way you can easily restore the original files if you find that deleting them doesn't help matters.

Tweak KDE for Better Performance

KDE is very similar to GNOME in overall features, so degunking and fine-tuning KDE is quite similar to degunking and fine-tuning GNOME. You can begin with investigating KDE's configurable options, which you might want to tweak to your liking. As with GNOME, a major KDE component is its file manager, which you can configure to perform as you like. You should also keep your overall desktop uncluttered. If your KDE configuration becomes seriously unstable or unusable, you may want to look into deleting your configuration files to start anew.

A Tour of Adjustable KDE Features

Figure 4-6 shows a KDE 3.3.1 desktop on a Fedora 3 system. If you compare this figure to Figure 4-1, you'll see quite a few similarities. Some are the result of Fedora using similar themes and desktop configurations for GNOME and KDE. Others are the result of more fundamental similarities between GNOME and KDE. You'll also notice some differences between the GNOME and KDE screen shots, and these tend to become more important the deeper you look.

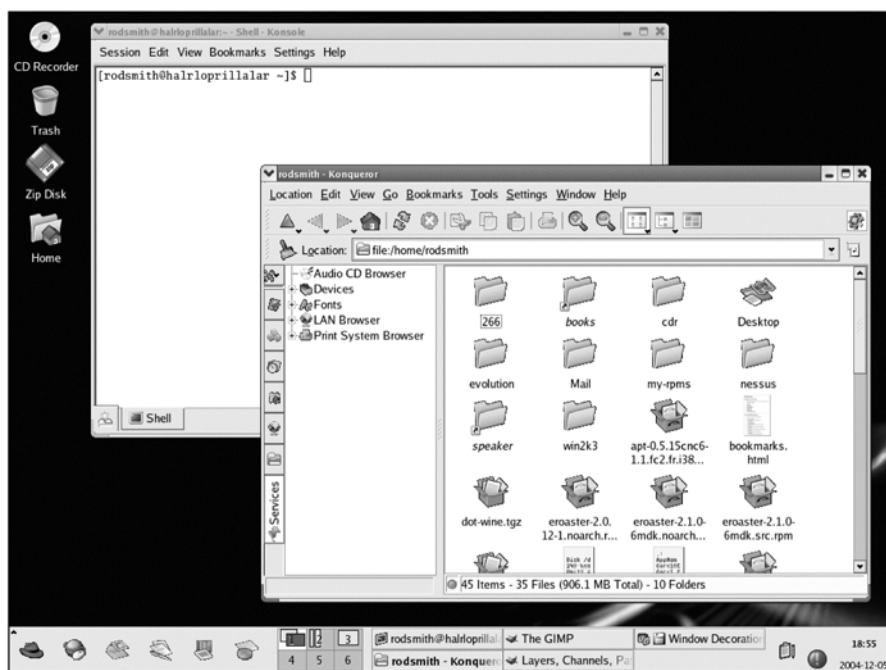


Figure 4-6

KDE provides features similar to those of GNOME.

NOTE: Some distributions present a wizard that enables you to set some defaults the first time you launch KDE.

Many of KDE's most important features can be set from the KDE Control Center, which can usually be launched from the main KDE menu—the red hat in the lower-left corner of Figure 4-6. (This icon is a stylized *K* in unmodified KDE configurations, but distributions often change it to their own logo.) The result resembles Figure 4-7. You select options you want to adjust from the index list on the left side of this window. In response, the contents in the right side of the window change, enabling you to set the options you want to set. General categories you can adjust are as follows:

- ✓ **Appearance & Themes**—The Appearance & Themes section provides options relating to the appearance, but not the functionality, of the desktop. These include background images, color schemes, fonts, icons, a screen saver, window borders, and themes. As with GNOME, you can install themes you obtain from the Internet. (The KDE Control Center has a button you can click to launch KDE's Web browser on a site with KDE themes.)
- ✓ **Desktop**—This section provides options related to the desktop and file browsing. You can adjust the number of virtual desktops, the number and placement of Panels, what file types generate previews in the file manager, and so on.

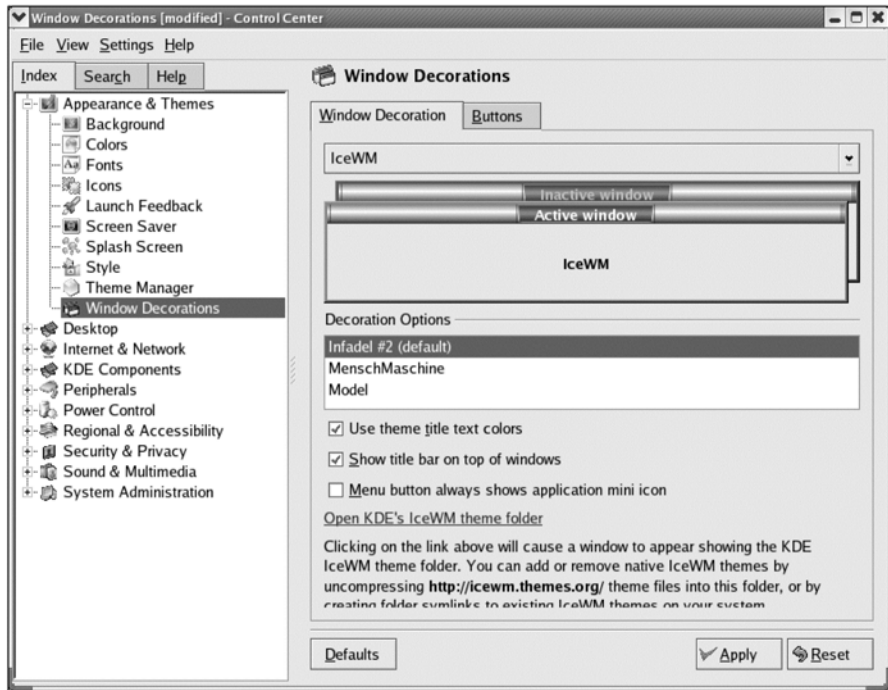


Figure 4-7

The KDE Control Center provides the means to adjust many of KDE's default behaviors.

- ✓ **Internet & Network**—You can adjust various network options from this area. These options relate to the way network clients, such as mail readers and Web browsers, interact with the network. You can also activate desktop sharing, which enables people using other computers to view and even use your own desktop from a remote location. (This can be handy for doing remote presentations.) Some of these features are described elsewhere in this book.
- ✓ **KDE Components**—This area provides options related to KDE's component programs. Some of these features relate to the way KDE programs communicate with one another. For instance, you can specify a mail reader that should be launched by any KDE program that wants to launch a mail reader. Of particular interest from a degunking point of view is the KDE Performance sub-area. Here you can set features that influence KDE's performance, such as whether it tries to optimize itself for low-memory configurations or try to *preload* (that is, load before it's called) critical code. You should experiment with these options; depending on your available RAM and how many programs you have running, you might find that one setting or another will improve performance.

- ✓ **Peripherals**—You can set options relating to hardware peripherals in this area. These options include the screen resolution, keyboard repeat rate, mouse tracking speed, and the default printer. These options do *not* affect the low-level driver configuration, though; for that, you need to use administrative tools rather than user tools.
- ✓ **Power Control**—Particularly if you're using a laptop computer, you should check out this area, which enables you to set various power-saving options and features that are most useful on laptops.
- ✓ **Regional & Accessibility**—You can change the localization for your keyboard, set options intended for those with reduced manual dexterity, and so on from this section.
- ✓ **Security & Privacy**—This section provides options related to security and privacy, including a password-change tool and the ability to enable encryption and digital signature features in various KDE programs.
- ✓ **Sound & Multimedia**—You can tell KDE about your sound system, enable sounds to be associated with specific events in particular KDE programs, and so on in this area. This feature doesn't affect low-level sound drivers, though. If you can't get sound to work at all in your programs, you may need to deal with the low-level sound drivers rather than the KDE sound configuration.
- ✓ **System Administration**—This area provides a way to perform some common system administration tasks, such as installing fonts and changing the login configuration. Some of these settings require administrative access and prompt you to click a button and type the root password before you can make changes. Others, such as font installation, can work in your home directory alone.

Most of these features apply to KDE applications *only*. For instance, if you set a default font or a sound event, that setting will apply only to KDE programs; other programs will be unaffected. This fact is particularly important for the security features. Telling KDE to use a Web proxy server, for instance, will affect KDE Web browsers such as Konqueror, but will not affect non-KDE Web browsers, such as Mozilla Firefox. Changing your login password using the KDE tool affects your login whether you use KDE or some other desktop environment, though.

Most KDE configurations use a single Panel, at the bottom of the screen, unlike the two that are more common with GNOME. KDE's default Panel includes features much like those in GNOME, though, including launchers for programs, a pager, buttons to bring specific windows to the front, a clock, and so on. You can configure the Panel as a whole by right-clicking in a bare part of it, or you can configure a specific component by right-clicking it. For instance, right-clicking the pager enables you to change the number of virtual desktops KDE displays, alter the information about each virtual desktop that's shown in the pager, and so on.

Tweaking File Browsing in KDE

KDE's file manager is Konqueror. Earlier, I mentioned Konqueror in the context of Web browsers, and in fact, the program handles both tasks. If you enter a local directory specification in Konqueror's Location field, it acts as a file manager. If you enter a Web site address, the program functions as a Web browser. The fact that the KDE file manager is also a Web browser means that you can have fewer programs running, but it also adds some degree of bloat to the file manager side of things, particularly if you prefer to use another program as your Web browser.

Konqueror's double duty also means that configuring the file manager side can be a more daunting task than configuring GNOME's Nautilus file manager. You can access the main Konqueror configuration tool from the Settings > Configure Konqueror menu in a file manager window. The result resembles Figure 4-8. This dialog box breaks Konqueror options into broad categories along the left side of the window. Select one, and options you can set appear on the right side of the window. The main categories that are most important for Konqueror's file management duties are as follows:

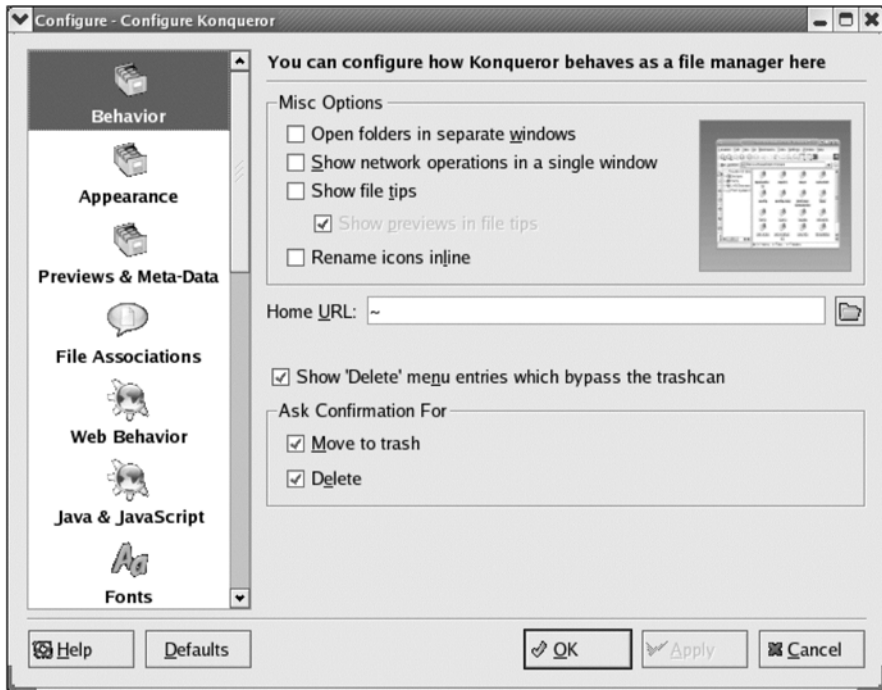


Figure 4-8

Konqueror's Configure dialog box provides an unusually wide array of options for a file manager.

- ✓ **Behavior**—This area contains options that affect how Konqueror reacts to your actions—whether opening a folder creates a new window or devotes the existing window to the new folder, whether file information appears when you move the mouse over the file (the so-called “file tips”), whether Konqueror asks for confirmation when you empty the trash, and so on.
- ✓ **Appearance**—You can set a few appearance options here, such as what font Konqueror uses.
- ✓ **Previews & Meta-Data**—This section has options related to previews. You can enable network and local protocols and file types that should support previews and set a few other options, such as limiting previews to files that are smaller than a size you specify. (This section is called Previews in some versions of Konqueror.)
- ✓ **File Associations**—From this area, you can link filename extensions to file types, and in turn link those to icons and applications that can be used to access the files.

Options below these in Konqueror’s Configure dialog box relate to its Web browser duties, so you needn’t be concerned with them if you’re only interested in getting the program working as a file manager.

In addition to these settings, you can adjust some Konqueror features through its menu options. In particular, the View menu provides options that relate to its display of files and directories—how large icons should be; whether to display icons with filenames alone or to display fuller listings that include file sizes, ownership, and other features; whether to use a solid background color or use an image as a background, and so on.

As with most Nautilus options, Konqueror’s options are largely a matter of personal preference. A few options can impact performance, though, and so can be considered more objectively gunky. In particular, file previews can become tedious if your hard disk is slow, if files are particularly large, or if a directory contains dozens or hundreds of files that must be previewed. If these conditions don’t apply, though, previews can be a nice convenience.

Keeping Your KDE Desktop Uncluttered

A default KDE configuration tends to use relatively few desktop icons. Figure 4-6 shows just four, and two of those (CD Recorder and Zip Disk) are for external disk devices. Your system might not have the same icons. Some distributions load up more icons in their KDE desktops, and you can do the same. Right-click an empty part of the desktop and check out the options under the Create New option in the resulting context menu. You can create a folder, a file, or a device. The last two options are actually links to submenus, each with its

own options. Files include common document types and programs, while devices refer to external storage devices, such as Zip disks and floppy disks.

As with GNOME desktop clutter, managing KDE desktop clutter is a matter of organization and self-control. Having direct links to a few frequently used programs, files, and directories makes good sense. Creating links to every minor program or file will only serve to create an unmanageable mess. As a rule of thumb, try to keep desktop icons down to a dozen or so. If necessary, place groups of related icons in folders to minimize the number that are constantly visible on your main desktop.

Starting with a Fresh KDE Configuration

KDE configurations can, like those for other programs, become so mangled as to be unusable. This can happen if you experiment too much with KDE options, rendering them a useless mishmash. This can also happen if you switch from one Linux distribution to another—many of the icons, file locations, and so on embedded in the first distribution's configuration files can become useless on the new one.

To correct such problems in a radical way, you can delete the KDE configuration files. (Be sure to do this when KDE is *not* running, though. If you don't, the running KDE instance may become confused and programs may crash.) When you delete its configuration files and restart KDE, it will create fresh configuration files that reflect the defaults for your system. Chapter 3 describes this process in general, and Table 3-2 in that chapter lists many common configuration files that can be deleted to restore default settings. For KDE, the most important of these is `.kde`, which is a subdirectory in which configuration files for many KDE programs appear. Sometimes deleting the `.qt` subdirectory is also important; this subdirectory holds configuration files used by Qt, which is a development library upon which KDE is built.

As when deleting any configuration file, the safest way to handle this task is to move the files or directories into a special directory or rename the files. This way, you can easily recover if you discover that you've made a mistake. The `.kde` directory holds options you might not want to lose, such as the address book used by the KMail mail reader. Wiping out such files by accident can be a real problem!

Use Alternative Desktop Environments

GNOME and KDE are both popular desktop environments with lots of features. Users who migrate from Windows should find both environments reasonably comfortable. Unfortunately, the full-featured nature of both environments can

be a real problem because all those features increase the size of the environments, both in terms of disk space consumed and RAM needed. Some features, such as file previews in file browsers, can also consume a fair amount of CPU time. Thus, some people prefer to use slimmer environments—GNOME and KDE are just too hopelessly gunky for these people's tastes. You're particularly likely to agree with this position if your computer is relatively low-powered—say, less than 256MB of RAM or a CPU that's weaker than a 1.5GHz Pentium 3.

The following pages describe three possible alternatives to GNOME or KDE: a slim desktop environment known as *XFce*, using a bare window manager, and creating your own desktop environment. I also describe how to tell your system to use your trimmed-down desktop environment.

Using XFce

XFce (<http://www.xfce.org>) is designed as a lightweight desktop environment. Unlike GNOME and KDE, which borrow heavily from Windows, XFce is loosely modeled after the Common Desktop Environment (CDE), which is a commercial desktop environment for Unix systems. XFce is open source, though. XFce bears some superficial resemblance to Mac OS X's environment, although XFce is much slimmer. In fact, it's XFce's small footprint that's one of its biggest drawing points.

On a freshly booted Fedora 3 system, starting GNOME consumes 44MB of RAM, while starting KDE chews up 26MB of RAM. (Both figures are compared to running nothing but the GUI login screen, which itself consumes a bit of RAM, so these environments actually use slightly more RAM than this.) One of XFce's great advantages is that it consumes just 9MB of RAM—roughly a third of what KDE uses and not much more than a fifth of what GNOME requires. While that difference may not seem like much on modern computers with half a gigabyte or more of RAM, it is important on smaller systems. A few megabytes might also be important on larger computers if you're running large programs. Furthermore, XFce launches more quickly than GNOME or KDE on the same hardware, even when that hardware is fairly quick.

So how does XFce manage to deliver the slimmed-down goods? XFce's main tool for doing this is simply providing fewer features—it doesn't provide desktop icons by default, its file manager is simpler, and so on. XFce does support the most important features, though, such as program-launch capabilities, a Panel, a task bar, and applets to change features like the keyboard repeat rate and the mouse tracking speed. The result looks something like Figure 4-9, which shows XFce 4.0.6 running on a Fedora 3 system. This configuration is

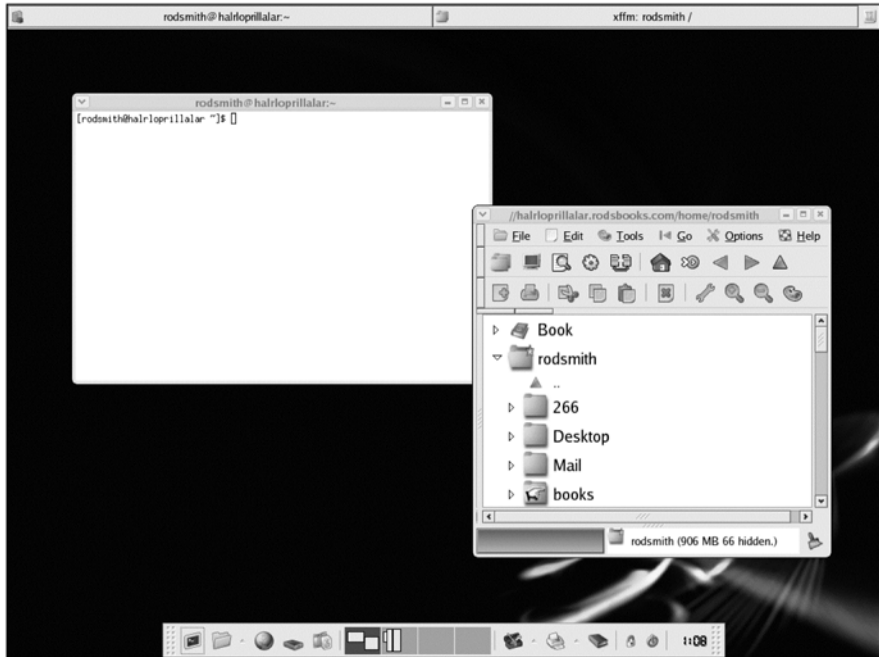


Figure 4-9

A default XFce configuration is somewhat simpler than a default GNOME or KDE configuration.

close to the XFce default, although Fedora provides a theme with the same desktop background and window borders that it uses with its GNOME and KDE configurations.

By default, a task bar that enables you to go directly to any open window appears at the top of the screen. The XFce Panel appears at the bottom of the screen. (Both can be moved using the XFce configuration tool.) Overall XFce configuration is done through the Settings Manager, which is shown in Figure 4-10. This tool provides options to control the desktop background image, the Panel, the mouse, the keyboard, and so on. These tools tend to be less elaborate than their GNOME and KDE counterparts, but they do provide the most important options.

Like the GNOME and KDE Panels, the XFce Panel supports a pager, Panel-mounted applets, and tools for launching programs. The XFce program launching feature, though, works a bit differently. Typically, each entry launches a primary application in a group, such as the Firefox Web browser in a Web browser group. You can often click on an arrow to the side of an item to obtain a list of additional options, such as the Konqueror and Opera Web browsers. You can then launch those other options by clicking their icons.

**Figure 4-10**

XFce's configuration tools tend to be simpler than those for GNOME and KDE.

GunkBuster's Notebook: Adding a Program Launcher with XFce

The default XFce configuration provides relatively few program launchers, and most distributions don't add much to this list—they put most of their effort into their GNOME and KDE configurations. Thus, if you want to use XFce, you may need to add program launchers. To do so, right-click an existing category and select Add New Item > Launcher from the resulting context menu. To add a new category, do the same from the very edge of the Panel, where there are no icons. The result is a dialog box in which you can type a command (that is, what you'd type at a command prompt to launch the program), select an icon, and so on. You can change an existing entry by right-clicking it and selecting Properties. The result is the same dialog box you used to create a launcher. For entries that are directly on the menu, you can check a box marked Attach Menu to Launcher in order to create a menu of related programs.

The XFce file manager is simple compared to Nautilus or, especially, Konqueror. It provides just a single view of the system and can seem a bit limiting, particularly if you're not used to it. If you find it inadequate, though, you can run another file manager. In fact, many default XFce configurations provide launchers for Nautilus and Konqueror, so you can use one of them if you prefer. Of course, doing so tends to reduce the benefits of running XFce rather than GNOME or KDE, but you might find that running a somewhat larger file manager is worth the cost in performance. If you use Konqueror as your Web browser, using it as a file manager might also make sense.

Overall, using XFce can be a liberating experience. It can make a fast computer seem even faster, and it can make an old system that's unbearably sluggish under GNOME or KDE bearable. Like living in a secluded cabin, though, XFce isn't for everybody. It achieves its speed benefits at the cost of numerous small creature comforts. If you're unwilling to relinquish those comforts, stick with GNOME or KDE. If you're curious, though, give XFce a try. (The upcoming section "Configuring Linux to Use Your Alternative" describes how to do this.)

Using a Bare Window Manager

An even more radical approach to slimming down your system is to use a window manager alone. Desktop environments are built around window managers, but they're much more than that. Even the slim XFce includes a Panel, applets to adjust the keyboard repeat rate and other features, a file manager, and so on. In years gone by, such programs were not commonly bundled together, so rather than select a desktop environment, you'd pick your favorite window manager. Today, you can still do this, and the benefit is further reduction in memory requirements compared to the benefit of switching from GNOME or KDE to XFce. Precisely how great a benefit you'll realize depends on the window manager you select. As an example, though, consider IceWM. On a Fedora 3 system, logging into an IceWM session actually *increases* the available RAM—IceWM consumes less memory than the GUI login program!

Clearly, the memory benefit of running a bare window manager (or at least a slim one like IceWM) is substantial, at least if you're running into memory problems created by the demands of GNOME, KDE, or XFce. On the other hand, bare window managers do make for a rather Spartan environment. Figure 4-11 shows an example: the IceWM 1.2.17 window manager running under Fedora 3. As with the GNOME, KDE, and XFce screen shots in Figures 4-1, 4-6, and 4-9, Figure 4-11 includes an open command prompt window (an `xterm`); however, because IceWM doesn't include its own file manager, Figure 4-11 doesn't show one. Of course, you can run a file manager in IceWM, or in any other window manager. In fact, part of the advantage of running a bare

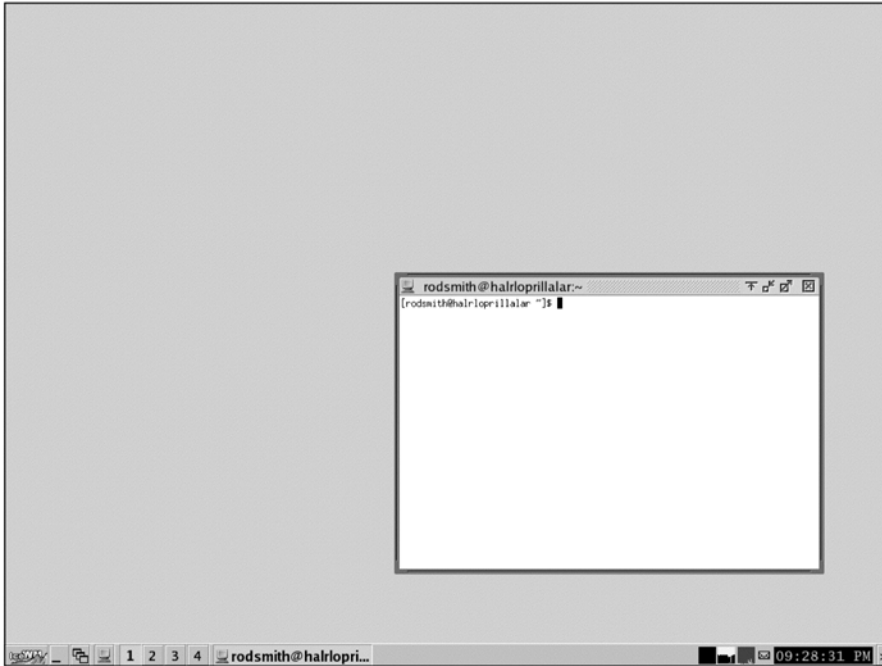


Figure 4-11

A bare window manager, such as IceWM, provides fewer features than a full desktop environment.

window manager is that you can pick precisely the components you want for other functions. (The next section, “Rolling Your Own Environment,” elaborates upon this advantage.)

Just what do you get from a bare window manager? That depends on the program you select. Some are *extremely* Spartan; they provide minimal functional controls for dragging and resizing windows, a simple tool for launching a handful of programs, and not much else. Others provide extra features, such as a task bar for selecting individual open windows, a pager for managing multiple virtual desktops, a clock, configurable themes so you can customize the look of the environment, and so on. Literally dozens, if not hundreds, of window managers are available for Linux. Here are some of the more common ones:

- ✓ **Blackbox**—Headquartered at <http://blackboxwm.sf.net>, this window manager is designed to be very low in resource use while still providing a good set of features.
- ✓ **Enlightenment**—Also known as *E*, this window manager is very large and feature laden, with an emphasis on extreme configurability and complex themes. It’s popular among those who like to tweak every detail of the look

of their desktop. You can learn more at <http://enlightenment.org/pages/main.html>.

- ✓ **FVWM**—This window manager, headquartered at <http://www.fvwm.org>, was the dominant Linux window manager in the mid-1990s, before GNOME and KDE came onto the scene. It provides a good set of features but is a bit resource heavy for what it provides. (On modern systems, though, saying that FVWM is resource heavy compared to other window managers is like saying somebody is a very poor millionaire.)
- ✓ **FVWM '95**—This window manager is an offshoot of FVWM. It adds a task bar and a new style modeled after that of Windows 95, which was current when FVWM '95 was developed. There is no FVWM '95 Web site, but you can download it from <ftp://mitac11.uia.ac.be/pub/> if it's not available with your distribution.
- ✓ **IceWM**—This is my personal favorite bare window manager. It provides a reasonable set of features, including a pager, a clock, and an assortment of themes for visual interest. It's also fairly slim in terms of resource requirements. You can learn more at <http://www.icewm.org>.
- ✓ **MWM**—The Motif Window Manager is most important because of its links to the Motif widget set, which is a popular set of programming tools for creating GUI applications. Motif has historically been used by commercial Unix and Linux programs, but most open source programs use GTK+, Qt, or some other widget set. In any event, Motif ships with MWM, and so MWM has often been used on commercial Unix systems.
- ✓ **TWM**—This window manager is something of a lowest common denominator. It's very old and provides very few features, even by bare window manager standards. It's almost always installed with Linux X servers and often functions as a fallback—if Linux can't start another window manager or desktop environment, it tries TWM as a last resort.
- ✓ **Window Maker**—This window manager aims to emulate the NeXT computer interface. As NeXT's interface was the basis for Mac OS X's interface, it shares some features with Mac OS X; however, Mac OS X has added to this feature set, whereas Window Maker implements only a subset of it. Still, Window Maker has a devoted following among bare window manager fans, and it's quite capable in this realm. Consult <http://www.windowmaker.org> for more information.
- ✓ **WM2**—This unusual window manager is very Spartan; it provides almost no features beyond those that are absolutely required of window managers. Its most striking feature, at least initially, is that it places window title bars to the *left* of the main window rather than above the main window as most window managers do. You can learn more at <http://www.all-day-breakfast.com/wm2/>.

This list is far from complete. As I said, there are dozens of window managers for Linux. Consult <http://xwinman.org> for a list of close to a hundred window managers. If you're interested in trying a bare window manager, you should probably try several of them. This will require installing the software, configuring your system to use it (as described shortly, in "Configuring Linux to Use Your Alternative"), and perhaps configuring the window manager to your taste. This last task is very specific to the window manager itself. Most have configuration dot files named after themselves. Many window managers don't create or modify these files, though; they use systemwide defaults if they don't find user configuration files. You should consult the window manager's documentation to learn how to create and edit its configuration files.

Rolling Your Own Environment

Desktop environments like GNOME, KDE, and XFce are really just collections of programs—window managers, file managers, applets to do a myriad of tiny tasks, and increasingly, larger applications to do not-so-tiny tasks. In principle, you can put together your own desktop environment by creating your own unique collection of programs to handle these tasks. For instance, you might combine the Blackbox window manager, the Nautilus file manager, the standard `xclock` desktop clock program, and so on. The result might have all the functionality you need in a desktop environment without all the gunk that's provided with the prepackaged ones.

Unfortunately, creating your own desktop environment isn't without its problems. The first is the sheer effort involved: You must research every single component you want, install the necessary software, and configure it all to start up or be easily launched from your window manager. This is a nontrivial undertaking that's best handled by those with considerable experience with Linux and its available software. The second major problem with this approach is that it doesn't support the sort of integration that a prepackaged desktop environment provides. The programs that ship with major desktop environments are designed to work together. For instance, programs that need it can share a common address book. They also respond to the in-application style and theme settings (for default application fonts, for instance) provided by the desktop environment. Chances are the programs you put together yourself will lack this sort of integration.

If you decide to try building your own desktop environment, start with the window manager. Pick one you like and learn how to configure it. You can then add other components by installing them and configuring the window manager's program-launching tool to launch the extras. If you want a tool to

be running at all times, you'll want to configure it to launch as part of your GUI login script, as described shortly in "Configuring Linux to Use Your Alternative."

Actually locating all of the extras can be a problem, particularly if you're not already very familiar with the Linux software scene. One trick you can use is to grab components from the major desktop environments. For instance, you might use the Nautilus file manager from GNOME or KDE's KMail mail reader. With few exceptions, applets and larger applications that are officially part of a desktop environment work just fine outside of that environment, although they may not integrate well with other tools. One problem with lifting components from a desktop environment, though, is that they may require a huge number of extra tools from the main desktop environment to work well. This extra baggage tends to gunk up your system, in that it can greatly increase the memory load of running what should be a fairly small program. As you try programs, monitor memory use. Type **free** at a command prompt and study the output, and especially the `-/+ buffers/cache` line, which shows the memory used and available *without* disk caches and buffers. If launching a small program causes a huge jump in memory used, it may be that the program is loading a bunch of extra gunk.

NOTE: Chapter 8, "Managing Processes," describes ways you can track down individual programs that are consuming too much memory.

For additional software ideas, you can consult an index of Linux software, such as the one maintained at <http://www.linux.org/apps/>. This site's GUI/Desktop and GUI/Applets categories are particularly relevant for the sorts of small utilities that typically make up a desktop environment.

Configuring Linux to Use Your Alternative

Most modern Linux distributions use GUI login tools that enable you to log into your preferred desktop environment. These tools usually consist of the GNOME Display Manager (GDM) or the KDE Display Manager (KDM) login program, although the original X Display Manager (XDM) and a couple of other alternatives also exist. In any event, these tools typically present fields for the username and password (GDM displays the password field only after you've entered a username; most others display both fields simultaneously). GDM and KDM also provide some way for you to select your login environment. For instance, Figure 4-12 shows a GDM login screen on a Gentoo Linux system. Before you finalize the login, click Session to see a list of environments, as

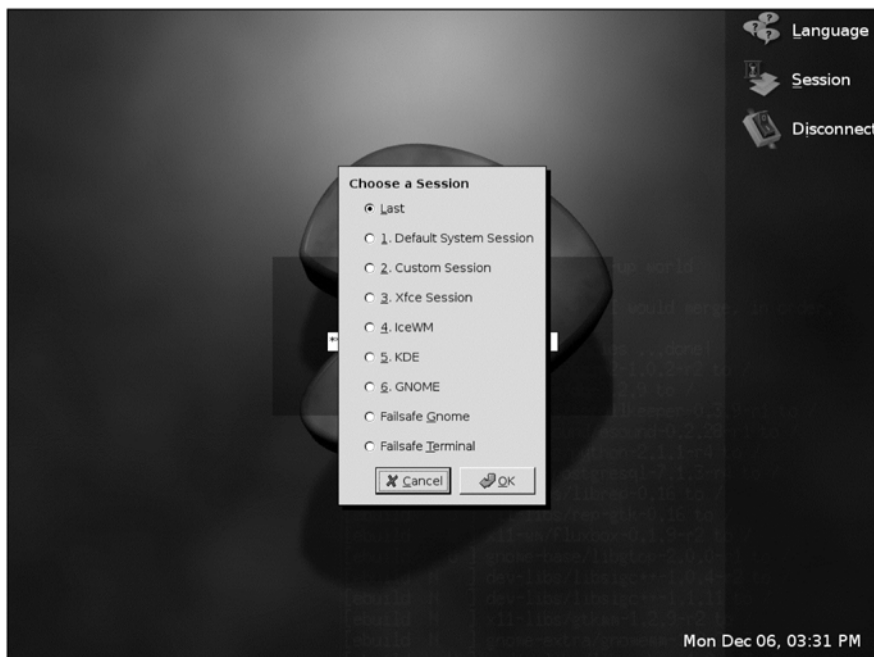


Figure 4-12

The GDM and KDM login tools provide a way for you to select your desktop environment or window manager when you log in.

shown in the figure. Click the one you want and then click OK to select that session type. KDM works in a similar way, but its options are available from a button called Session Type.

Most Linux distributions configure themselves so that you can easily select GNOME or KDE, at least if you've installed both environments. Accessing XFce is more hit or miss, and bare window managers are even less likely to appear in these menus. If you don't see your preferred environment as a login option, you have two choices for how to proceed:

- ✓ Select a session type that launches a user's X login script. This script is likely to be called `.xsession`, `.xsession`, or `.xinitrc` and is stored in the user's home directory. You can then configure this option as described shortly to launch your preferred programs. Unfortunately, this option has been disappearing from default GDM and KDM configurations, so it may not be present. If it is present, it may not have an obvious name.
- ✓ Add a new session type to your configuration. This new session type will run a systemwide script that launches whatever programs you tell it to run. Making this change requires root privileges.

If you follow the second path, you must first locate where your login program looks for its list of sessions. Most modern distributions use `/usr/share/xsessions` to store this information for both GDM and KDM. This directory holds files with `.desktop` extensions, such as `IceWM.desktop` to define an IceWM login session. These files provide information on the session, including its name and how to launch it. For instance, a file to launch a bare IceWM session might look like this:

```
[Desktop Entry]
Encoding=UTF-8
Name=IceWM
Comment=This session logs you into IceWM
Exec=icewm-session
TryExec=icewm-session
# no icon yet, only the top three are currently used
Icon=
Type=Application
```

The most important part of this file is the `Exec` line, which points to a session login script or launch program. (You should either provide a complete path to the file or ensure that the script is in a common program file directory, such as `/usr/bin`; otherwise, Linux won't be able to find it.) Desktop environments typically ship with programs that start up the rest of the environment, such as `startkde` or `gnome-session`. Some window managers do the same; for instance, `icewm-session` is part of the IceWM package.

If you want to do more than launch a single program for your startup procedure, you must create a script to do the job. This script takes the same form as an `.xsession` script in a user's home directory if your login system supports this option. Such scripts are fairly easy to create as Linux scripts go. They begin with a line that identifies them as scripts, and then contain a single line for each program you want to launch. For instance, suppose you want to run IceWM as your desktop environment, but you also want to launch the Nautilus file manager and an `xterm` window. Your script would look something like this:

```
#!/bin/bash
xterm &
nautilus &
icewm
```

The first line identifies the file as a Bash script; you should include this line in any startup script. Subsequent lines each start one program: `xterm`, `nautilus`, and `icewm`, in this case. The ampersand (&) after the first two of these lines causes the script to continue execution. The last line lacks this detail, so the script stops running once IceWM is started. When IceWM terminates (because the user selects a logout option), the script continues execution and terminates itself. This ends the session and returns control to the GDM or KDM login program. If you omit the ampersands on either of the first two lines, the window manager won't start until you exit from the earlier program, and if you add an ampersand to the final line, the script—and hence the login session—will terminate before you can do anything with your session. Thus, you should be sure to get the ampersands right!

Once you create this file and place it somewhere convenient (such as `/usr/local/bin` for a systemwide file, or this might be your `.xsession` file), you must mark it as executable. You do this with the `chmod` command, which changes the permissions (or *mode*, hence the name, which is short for *change mode*). Specifically, you want to add execute (x) permissions for all (a) users:

```
# chmod a+x my-login-script
```

You can then refer to the login script in your session file, if it's a global script. Log out of any GUI session you're currently running and restart the GDM or KDM session. In most cases, you can do this by logging into a text-only session and typing `telinit 3` followed by `telinit 5`. This changes the Linux *runlevel*, and as most Linux distributions run the GDM or KDM program only in runlevel 5, the effect is to shut it down and then start it up again. Some distributions, though, such as Debian and Gentoo, start the GUI login program in another way. For them, you may need to type `/etc/init.d/xdm restart`. (On Debian, you may need to replace `xdm` with `gdm` or `kdm`, depending on which program you're using.) If you have problems with this procedure, rebooting the computer will do the trick, but that's a rather radical solution.

Once this is done, you should see your new session type in the list of available login sessions. Some distributions configure themselves to change your default session when you select a new one. Others (including the popular Fedora) require you to run a special program or edit a configuration file to change your default. Fedora prompts you to do this, but it tells you to run a program (the Desktop Switching Tool) that may not be easily accessible in your new environment. You can type `switchdesk` to do the job; the result is a window from which you can select your default environment.

Miscellaneous Tricks for Improving a Linux GUI

No matter what desktop environment you use, you can employ several tricks to help improve your user experience and keep it gunk-free:

- ✓ **Pagers**—Pagers can be great tools for helping to degunk your desktop. If you want to simultaneously run several programs, you can run each in a separate virtual desktop or run a few in each desktop. For instance, you might run network-related programs in one virtual desktop while running a word processor and spreadsheet in another virtual desktop. This can help to reduce screen clutter, making it easier to switch between programs. On the other hand, you might be tempted to start too many programs this way, degrading performance because their needs will exceed your available RAM.
- ✓ **Keep fonts under control**—Fonts can be fun to play with, and in some cases having lots of fonts is a necessity. Unfortunately, too many installed fonts can cause problems. Finding the right font from a huge list can be difficult, and program performance can suffer, particularly if the program tries to preview fonts. Many fonts are installed systemwide, as described in Chapter 12, so consult that chapter to degunk system fonts. You can also install fonts in your home directory, in the `.fonts` subdirectory. The GNOME and KDE environments provide tools to help you do so. You should be cautious about using this feature, and if you do use it, remove any fonts that you install but don't use, lest they become gunk.
- ✓ **Simplify your styles**—Styles, window decorations, and desktop backgrounds all qualify as “eye candy”—they can look good, but they don't add anything good to your desktop diet. At best, they do no harm. At worst, they can consume system resources or be distracting. If you're running X at a low bit depth (16-bit or, worse, 8-bit), elaborate styles may look awful because your display won't support enough colors, so using a simple style may be preferable. On any but very old display hardware, you can increase your X color depth, as described in Chapter 12; however, on very old video hardware, this is likely to noticeably reduce the speed of the display.
- ✓ **Don't be afraid to experiment**—You can always delete your desktop environment's configuration files to start over from scratch, so you shouldn't be afraid to experiment with new settings. You may discover a way to improve performance or add some feature that you find very helpful. Of course, if you spend all your time experimenting, you won't have any time left to do real work!

TIP: If you don't want to risk ruining your current configuration, try creating and using a spare account for experiments. Chapter 9, "Account Degunking," describes Linux account management tools. Create an account for the express purpose of trying new features or desktop environments. If you like what you see, implement the changes in your regular account. If you don't like the results, your main account is unharmed. Either way, remember to delete the account, lest it become gunk!

Summing Up: Keeping Your Desktop Environment Gunk-Free

Linux provides an unusually wide range of options in desktop environments. GNOME and KDE are both common, while XFce and various other tools are also available. Because these environments vary in their overall number of features and resource requirements, one excellent way to degunk a Linux desktop environment is to pick the right one for the job. If GNOME feels like an elephant on roller skates, switch to XFce or a bare window manager. If a bare window manager feels like a plain white room with no furniture, switch to XFce. Of course, within each desktop environment, you can also take steps to degunk it. All desktop environments provide configuration tools to adjust the mouse tracking speed, change the look of window borders, and so on. Thus, you can set all sorts of features that change the gunk level to suit your needs.



User Settings for Common Applications

Degunking Checklist:

- ✓ Clean up OpenOffice.org by installing it properly, managing your fonts and printers, and understanding the compatibility between OpenOffice.org files and other applications' files.
- ✓ Degunk Mozilla Firefox by managing its fonts and tweaking various security-related options.
- ✓ Keep the Evolution mail reader clean by organizing your e-mail folders, setting appropriate security options, and setting sensible HTML options.
- ✓ Know when to use alternative applications to help support your unique requirements.

The Linux OS is just a means to an end, the end being getting productive work done. That's accomplished with the help of application programs that run in Linux. These programs, like Linux itself, can easily accumulate gunk. Covering how to degunk every Linux application program would bloat this book to the size of an encyclopedia, so instead, I'll present an overview of just three common and powerful Linux applications: the OpenOffice.org office suite, the Mozilla Firefox Web browser, and the Evolution mail reader and contact manager.

Although this chapter is focused on showing you how to degunk these three programs, you can apply the general principles you learn here to other programs that handle the same tasks performed by these applications. Some of these principles can also be applied to many other types of programs. For instance, you might be able to apply some of what you learn about fonts in this chapter to graphics programs.

Degunk OpenOffice.org

The first step in this tour of applications is OpenOffice.org (<http://www.openoffice.org>). This office suite includes a word processor (Writer), a spreadsheet (Calc), a presentation program (Impress), a vector graphics program (Draw), and assorted utilities that help tie everything together. OpenOffice.org is the open source sibling of Sun's StarOffice (<http://www.sun.com/software/star/staroffice/>), which offers the same set of features plus a few extras, such as more import/export filters. Most of what I write about OpenOffice.org in this chapter applies equally well to StarOffice.

OpenOffice.org gunk can start to build up when you install it, so you'll want to be mindful of how you accomplish this task. OpenOffice.org can also suffer from font gunk. The office suite uses its own font rendering system, so knowing how to install and manage fonts is important. You can also define printers using the same tool, although OpenOffice.org usually auto-detects your printers correctly. In day-to-day use, you may need to know how to deal with files created in other programs (most notably the Microsoft Office suite that's popular on Windows and Mac OS). Finally, OpenOffice.org is often overkill, so I'll describe some of the slimmer alternatives to this package.

OpenOffice.org Installation Options

If your Linux distribution doesn't ship with OpenOffice.org, you have two choices for how to proceed:

- ✓ **Install OpenOffice.org from another distribution's packages.** You can locate OpenOffice.org package files for another distribution and install them on your own. This approach works best if your distribution uses RPM Package Manager (RPM) files. You can look for OpenOffice.org packages for another RPM-based distribution, such as Fedora, Mandrake, Red Hat, or SuSE. This approach isn't guaranteed to work because there might be differences in support libraries or other package conflicts. If it does work, it's usually the best way to proceed because RPMs are easy to install.
- ✓ **Install the package from the OpenOffice.org site.** When you download OpenOffice.org from its official Web site, you can select your preferred language, which is one advantage of this installation method. Another is that you can better fine-tune the specific components installed on your system. The file you download is a tarball—`00o_1.1.4_LinuxIntel_install.tar.gz` is the latest stable version for x86 systems as I type. (A 2.0 version may be available by the time you read this, though.) The OpenOffice.org tarball, when unpacked, contains a GUI installer program similar to those that are common on Windows. It presents several options, some of which can produce rather gunked-up installations.

Installing an RPM or other distribution-provided package usually produces a fairly gunk-free installation. The GUI procedure, though, is a bit weird by Linux standards, and you must select some fairly nonobvious options to produce an installation that's as clean as possible.

To begin, uncompress the tarball you downloaded by typing `tar xvzf 00o_1.1.4_LinuxIntel_install.tar.gz` at a command prompt in the directory in which you've stored the tarball. (Change the filename as necessary for the version you actually download.) This action creates a subdirectory that holds a large number of files, including the `install` script and the `setup` program. Why the developers had to put two programs (both with names that suggest they're used for installation) in the tarball I'll never know. Furthermore, simply running either of these commands with no options performs a single-user installation—the software is installed in the user's home directory. If several users want to run the software, it must be installed separately for each user. Because the OpenOffice.org installation is about 200MB in size, this can add up to a *lot* of gunk!

The cleaner solution is to perform what the OpenOffice.org developers refer to as a *network installation*. You can do this even if your system isn't connected to a network. This approach is similar to the way most software is installed in Linux. To do a network installation, follow these steps:

1. Acquire root privileges in a GUI terminal (such as an `xterm` window) by typing `su` and entering the root password.
2. Type `./setup -net` to run the setup program and perform a network install. (If you get a message complaining that the program couldn't connect to the X server, drop back to a normal user window and type `xhost +` to disable access controls on the X server. You can then start over from step 1.) The system may seem to do nothing for a few seconds, but then a GUI installation program should appear.
3. Follow the prompts in the installation program. For the most gunk-free installation, select a custom installation when asked for the installation type. The result is a dialog box like the one shown in Figure 5-1. Here you can select which components to install, so you can omit parts of the package you know you'll never need. For instance, Figure 5-1 shows the help for the Writer component and the TGA Import filter being omitted.

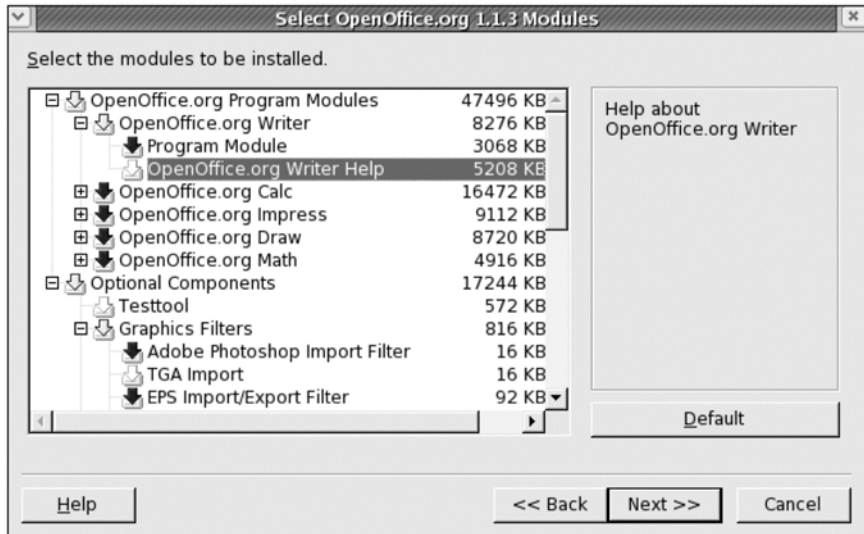


Figure 5-1

OpenOffice.org's installer enables you to select which components you want to install.

4. The installer asks for an installation directory. By default, this is `/opt/OpenOffice.org1.1.4` (or something similar for your version of the program). This is usually a reasonable default, but you might prefer putting the program in `/usr/local` or some other subdirectory.

Once the program is installed, users can launch it. Depending on your installation method, appropriate launchers may exist in your desktop environment. If not, or if you prefer to launch OpenOffice.org from a command prompt, you can

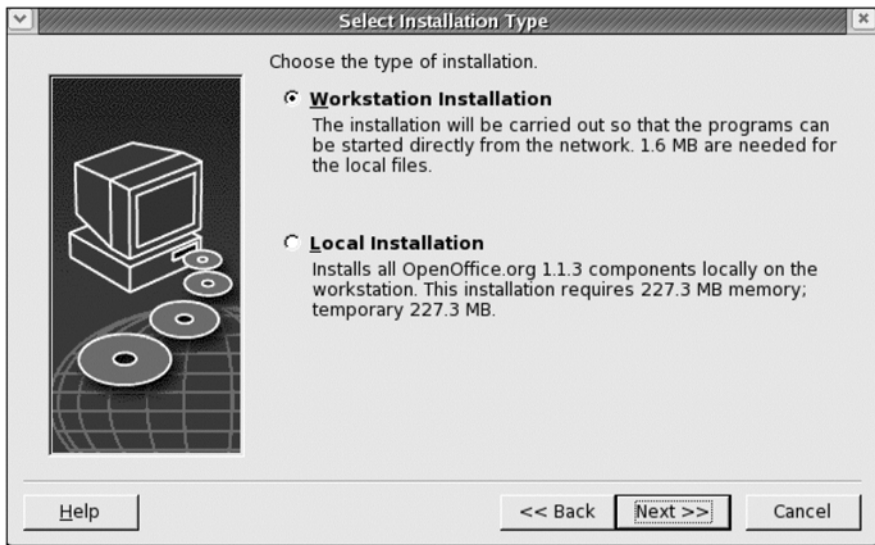
type the name of a component, such as `swriter` for Writer, `scal` for Calc, and so on. (Some distributions change these names to begin with `oo` rather than `s`. The `s` names are a holdover from OpenOffice.org's StarOffice heritage.) These program files are located in the program subdirectory of the OpenOffice.org installation directory. Unfortunately, this directory isn't likely to be on your shell's search path, so you'll need to do one of three things:

- ✓ **Type the complete pathname.** You can type the complete path to the program, such as `/opt/OpenOffice.org1.1.4/program/swriter` rather than simply `swriter`.
- ✓ **Add the directory to your path.** You can modify your `$PATH` environment variable to include the appropriate directory. This environment variable is usually set globally in `/etc/profile` and can be set for individuals in their `.bashrc` files. Be sure to locate an existing line that sets or modifies the `$PATH` variable and add the directory to that existing line.
- ✓ **Create symbolic links.** You can create symbolic links in a directory that's on your path (`/usr/local/bin` is usually a good choice). Change to this directory and type `ln -s /opt/OpenOffice.org1.1.4/program/swriter ./` to create a link for `swriter`. (Change the source directory as appropriate, of course.) Repeat this step for each program.

When a user launches OpenOffice.org for the first time, the installation program appears again! The reason is that OpenOffice.org needs to gather some information from users for their personalized sessions, such as their real names. During the process, OpenOffice.org also installs configuration files in the user's home directory. Some of these files consist of links to files in the main OpenOffice.org installation directory.

WARNING! During user setup, OpenOffice.org presents the *Select Installation Type* dialog box shown in Figure 5-2. Users should select the *Workstation Installation* option, **not** the *Local Installation* option. The latter option copies all of the OpenOffice.org program and support files to the user's home directory—a very gunky approach indeed.

After completing the user setup, users can launch OpenOffice.org components normally to use them. Because this book is about degunking, I won't describe most aspects of day-to-day OpenOffice.org use. Suffice it to say that the program works much like other office suites. You can write letters, manage your budget, create presentations, and so on.

**Figure 5-2**

Users should select the Workstation Installation option to avoid duplicating all the OpenOffice.org program files in their home directories.

Managing OpenOffice.org Fonts

As of version 1.1.4, OpenOffice.org uses its own fonts rather than those provided by X (and as described in Chapter 12, “Optimizing Your X Configuration”). Thus, if you want to use your own fonts in OpenOffice.org, you must install them in OpenOffice.org. Likewise, if you want to remove fonts to degunk your font menus, you must do so in OpenOffice.org.

NOTE: Some Linux distributions, such as Fedora, reconfigure their OpenOffice.org programs to look in system font directories for fonts. These modified OpenOffice.org programs lack the font installation options described here. Instead, you should manage your system fonts as described in Chapter 12 and the OpenOffice.org fonts should mirror those changes.

OpenOffice.org can use either TrueType or Adobe Type 1 (aka Adobe Type Manager, or ATM) fonts. All major Linux distributions ship with a wide variety of fonts of one or both types. These are usually located in subdirectories of `/usr/share/fonts` or `/usr/X11R6/lib/X11/fonts`. Check these directories to see what’s available on your system. TrueType fonts have filename extensions of `.ttf` and Type 1 fonts have extensions of `.pfa` or `.pfb`. Most other filename extensions denote other font types, such as bitmapped fonts, which OpenOffice.org can’t use. You can also download fonts from numerous sources on the Internet or buy CD-ROMs containing dozens or hundreds of fonts at most computer stores.

NOTE: Font filenames are often inscrutable. Don't worry too much about that, though. The OpenOffice.org font installer lets you preview fonts and refers to them by font name, rather than by font filename.

Once you know where the fonts you want to install are located, you can start the OpenOffice.org font utility. This program goes by the name `spadmin` or `spadmin.bin`. It's located in the same directory as the main OpenOffice.org program files. Launch this program and you should see a dialog box resembling the one shown in Figure 5-3. This main window is intended for managing printers, and I'll describe that function in a moment. First, though, click the Fonts button to manage fonts. The result is a dialog box similar to the one shown in Figure 5-4.

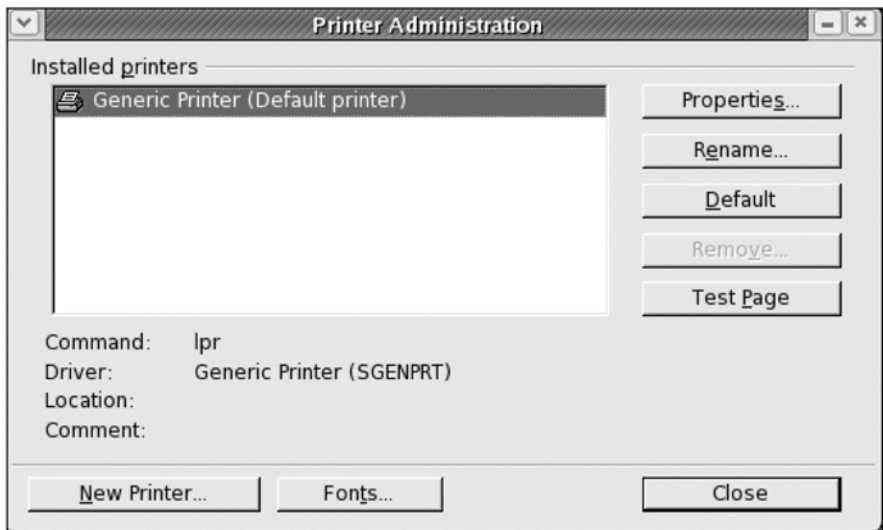
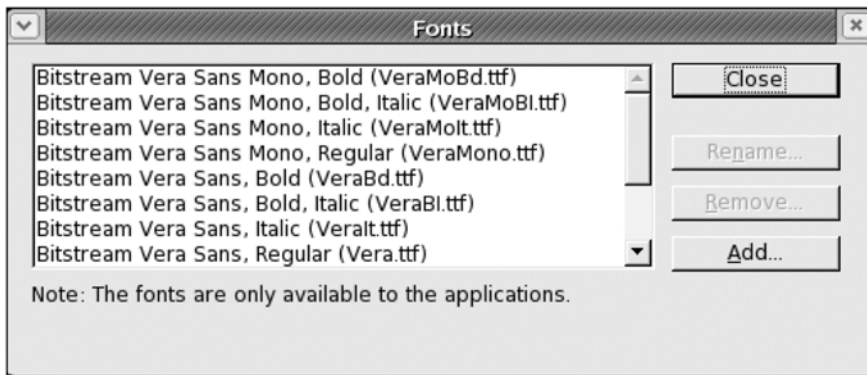


Figure 5-3

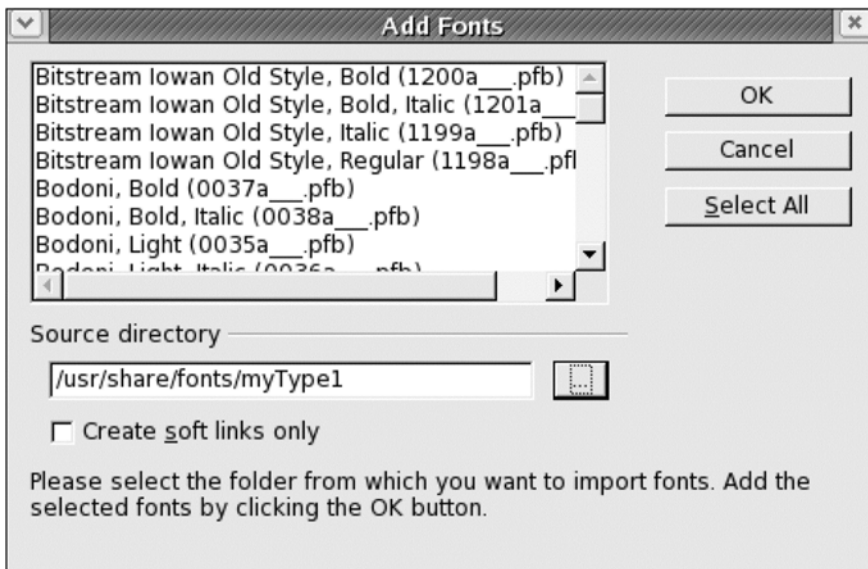
OpenOffice.org provides a special utility for managing fonts and printers.

NOTE: If you run `spadmin` as root, the changes affect all users of the computer. If you run it as an ordinary user, the changes affect you alone.

Once you're in the Fonts dialog box, you can begin managing them. If you want to remove fonts, select them and click Remove. To add fonts, click Add. The result is the Add Fonts dialog box shown in Figure 5-5. Click the ellipsis (...) next to the Source Directory field to locate the directory in which your fonts appear. (You can't type the name directly; it won't work.) The result is a list of fonts, including both their names and (in parentheses) the filename, as shown in Figure 5-5. Select the fonts you want to install. If you want to create

**Figure 5-4**

OpenOffice.org lists fonts by name, separating out each style.

**Figure 5-5**

The OpenOffice.org font installer displays fonts by name and by filename.

symbolic links to save disk space, check the Create Soft Links Only option. This option is good if the font is already installed on your system (say, for X) but not if you're installing fonts from a font CD-ROM. When you're done with this, click OK to install the fonts. The system should tell you that it's done so. You can then click Close in the Fonts dialog box.

Managing OpenOffice.org Printers

In most cases, OpenOffice.org detects your printers correctly. In Figure 5-3, only Generic Printer is listed as an option; however, if you select a print option from an OpenOffice.org component program, all the system's printers are listed. This detection works best when your system uses the Common Unix Printing System (CUPS), which is the preferred printing system with modern Linux distributions. If you find that OpenOffice.org doesn't detect your printers, but if you're sure that they do work from other Linux programs, you can force the issue by telling `spadmin` to add the printers. To do so, launch `spadmin` so that you see its main dialog box (Figure 5-3). You can then click **New Printer** to add a printer. The result is a series of dialog boxes that ask you about the printer. You must know its model (or a more popular but compatible model). The trickiest part of this process is specifying a command to print to the printer, as shown in Figure 5-6. In most cases, you'll enter an `lpr` command line that specifies the printer, such as one of the examples shown in Figure 5-6. (These examples are based on the printers defined on your own computer, as detected by OpenOffice.org. What you see won't exactly match Figure 5-6.) Sometimes, though, you might use something else. For instance, if you're using a peculiar utility to print in some unusual way or using an odd network protocol, you'd

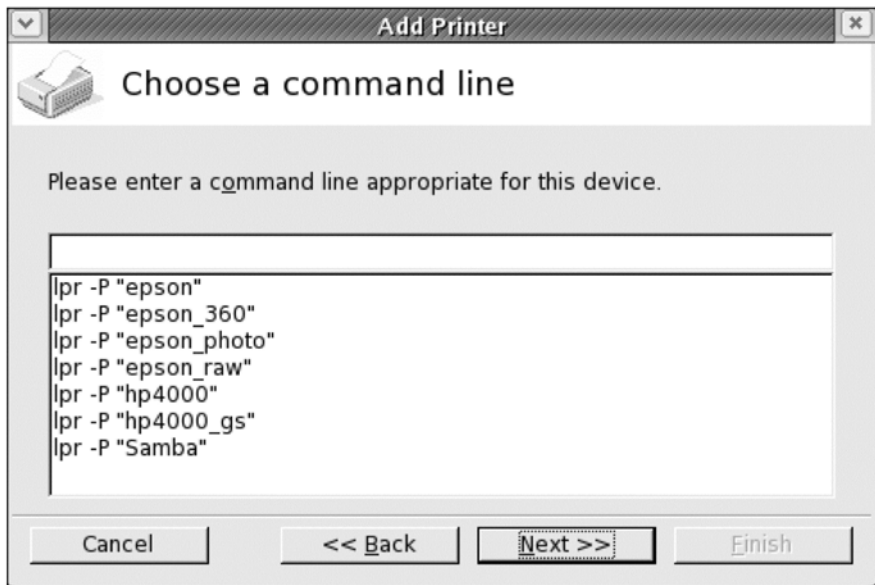


Figure 5-6

The OpenOffice.org printer installer needs to know how to print to any unusual printers you want to define.

enter another command. Precisely what this command might be, I can't say; you'll have to know enough about your local printing system to figure this out.

File Compatibility Issues

In the office suite world at large, Microsoft Office is the king of the hill. Many businesses and nonprofit organizations run on Microsoft Office as if it were gasoline; take it away and they'd come to a screeching halt. In many cases, if you want to do business with such an organization, you must be able to handle Microsoft Office documents. Fortunately, OpenOffice.org is about as good as they come in this respect—in fact, it's OpenOffice.org's compatibility with Microsoft Office documents, more than anything else, that makes it the office suite to beat in Linux.

That said, Microsoft's file formats are *not* OpenOffice.org's native formats. At its core, OpenOffice.org uses the Extensible Markup Language (XML) as its native document format. OpenOffice.org does something unusual, though: It takes several XML files, along with any necessary support files (such as graphics files for embedded graphics), and compresses them into a zip archive. The resulting files are saved with OpenOffice.org filename extensions, such as `.sxw` for Writer documents or `.sxc` for Calc documents.

Being the 400-pound gorilla of the office suite world, Microsoft can and does ignore the OpenOffice.org file formats. Thus, it's incumbent upon you, as a user of OpenOffice.org, to create Microsoft-compatible documents. This can be done from the usual OpenOffice.org file save dialog box; just select the appropriate Microsoft Office file format from the File Type field.

If you must exchange documents with Microsoft Office users, I recommend saving the file *both* in OpenOffice.org's native format *and* in the Microsoft Office format. Doing so ensures that you'll be able to read the original OpenOffice.org file if something goes wrong with the export process. (Such problems are rare in my experience, but it's better to be safe than sorry.) As far as reading Microsoft Office files goes, just click the filename from the file open dialog box. OpenOffice.org should be able to detect the file type and process it appropriately.

Although OpenOffice.org provides excellent Microsoft Office file format compatibility, its ability to handle other document types is more limited. You can't read or write WordPerfect files, for instance. Thus, if you must exchange documents with users of programs that create such files, you may need to use another program or agree upon some common file format, such as Microsoft

Office formats. (This is one area where the commercial OpenOffice.org variant, StarOffice, is superior to its open source counterpart.)

Overall, OpenOffice.org's file format compatibility with Microsoft Office is excellent. Still, problems can crop up. Fonts may not be identical, and precise layout features, such as line and page breaks, may not match. These differences can accumulate if a document is passed back and forth over time. If you have the luxury of doing so, your best bet is to keep your documents simple. If you have a choice, avoid using features such as tables and footnotes. If you must use these features, try to keep them as simple as possible. These steps will minimize the chance of gunk working its way into your documents.

Faster Alternatives to OpenOffice.org

OpenOffice.org is probably the most popular Linux office suite, but it's not the only one. Other suites, or other stand-alone programs that might be suite components, are available. Most of these programs are substantially smaller than OpenOffice.org, which makes them preferable on systems having less memory, CPU power, or disk space than whatever model appeared in stores yesterday. These programs can also be less intimidating to users with limited needs and experience. For all of these reasons, you may want to look into these programs as alternatives to OpenOffice.org:

- ✓ **KOffice**—This office suite is part of KDE. It includes a word processor, a spreadsheet, a presentation program, a vector graphics program, a bitmap graphics program, and assorted other smaller programs. For the most part, KOffice isn't as powerful as OpenOffice.org, but KOffice does offer more components than OpenOffice.org. You can learn more at <http://www.koffice.org>.
- ✓ **GNOME Office**—Like KDE, GNOME offers an office suite, headquartered at <http://www.gnome.org/gnome-office/>. This suite features the Abi Word word processor, the Gnumeric spreadsheet, and the GNOME-DB database. These programs were not originally written as a single suite, so they differ from one another more than the component programs in OpenOffice.org or KOffice do. Abi Word is notable for having an unusually complete set of import/export filters, although they don't do as good a job with Microsoft Office documents as OpenOffice.org Writer does.
- ✓ **Siag Office**—This small office suite features the Pathetic Writer word processor, the Siag spreadsheet, and the Egon Animator animation program. This suite, although small, sports some unusual features, such as a simple Web server in the spreadsheet component that enables you to deliver the spreadsheet as a Web page. You can learn more at <http://siag.nu>.

- ✓ **WordPerfect**—The once dominant word processor is available for Linux, although it's not seen significant updates in years. You can find a freely available version of WordPerfect 8 for download (check <http://www.linuxmafia.com/wpfaq/downloadwp8.html> for a list of sites). A pay version was once available and was rereleased briefly in mid-2004, but is now hard to find; eBay may be your best bet. The WordPerfect 2000 Office suite was also released for Linux, but this version used WINE to run Windows binaries under Linux. Getting this version running on modern distributions is nearly impossible.
- ✓ **TeX, LaTeX, and LyX**—Old hands at Unix are often familiar with TeX, which is a typesetting language famous for its capacity to handle layout of mathematical equations. LaTeX is an expanded version of TeX, and LyX (<http://www.lyx.org>) puts a GUI front end on LaTeX. These programs aren't good choices for the average person who's comfortable with conventional word processors, but technical users (mathematicians, scientists, and engineers) often like them.
- ✓ **DocBook**—Like TeX and LaTeX, DocBook is a text-based typesetting language. It's built on top of XML or the Standard Generalized Markup Language (SGML), and it's intended as a tool to help write technical documents. A great deal of Linux documentation is written in DocBook and then converted into a variety of formats, such as the Hypertext Markup Language (HTML) for Web pages, Portable Document Format (PDF) files, and so on.

For general-purpose use by relatively nontechnical users, KOffice and GNOME Office are both likely to be reasonable alternatives to OpenOffice.org. Siag Office might be worth investigating if your needs are particularly modest and your system resources are equally slim. WordPerfect is perfect if you need to exchange documents with other WordPerfect users, but its age means you'll have to jump through some extra hoops installing old libraries. It's also not as pretty as most modern programs. TeX, LaTeX, LyX, and DocBook are all useful mainly for technical users. Although they can be used by nontechnical people, such users are likely to find them strange at best and quite difficult to master at worst.

Degunking Mozilla Firefox

Mozilla (<http://www.mozilla.org>) is the open source spin-off of the Netscape (<http://www.netscape.com>) Web browser. For several years, Mozilla has become a very powerful but (to be honest) gunky Web browser. In 2004, though, Mozilla spawned a lighter variant, known as Firefox. In the first few months of its life, Firefox has attracted a lot of attention, and not just in the Linux world. (Firefox is also available for Windows and Mac OS X.)

Although Firefox is a nimble Web browser and is rapidly becoming the browser of choice in Linux, it's not completely immune to gunk collection. I'll therefore describe some of its pitfalls, beginning with font issues. Most potential Firefox gunk is security related, though; *any* Web browser can be a security swamp, and knowing how to navigate this swamp will help. Finally, despite all the hype surrounding Firefox, it might not be right for you. Alternative Linux browsers do exist, so you should know a bit about them.

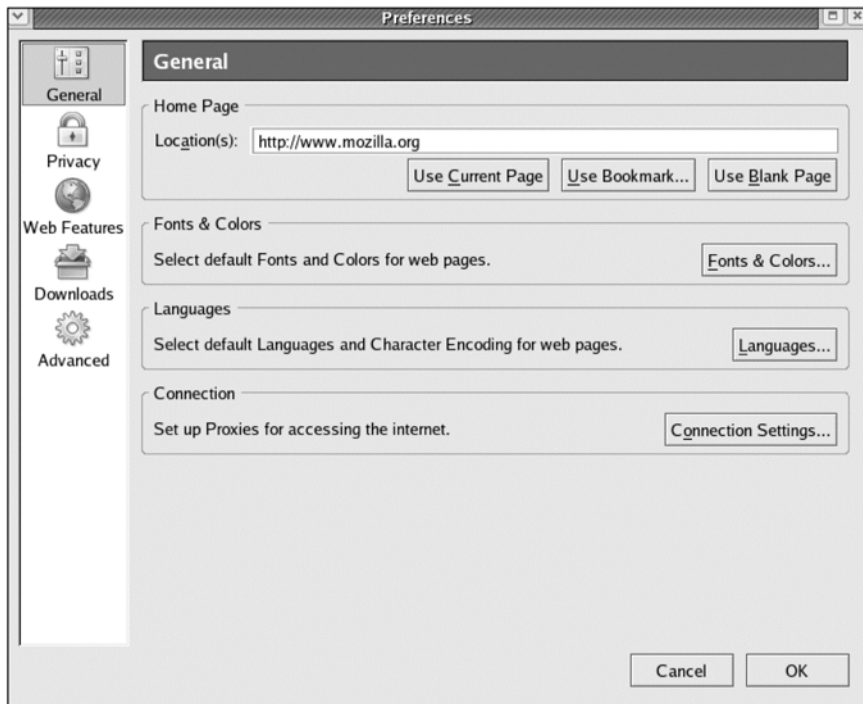
Improving Firefox's Fonts

One common complaint when using Linux for Web browsing is that fonts look bad. Sometimes the problem lies with the underlying font rendering system. Specifically, if your Web browser is using X core fonts rather than Xft fonts, the result is likely to be chunky fonts that may look rather unappealing, particularly if you use certain fonts or certain font sizes. To correct such problems, you should first ensure that Xft fonts are available on your system, as described in Chapter 12.

Beyond this issue, you should check the Firefox font options. These options, like most Firefox configuration options, are available from the Firefox Preferences dialog box, which you can reach by selecting Edit > Preferences. The result resembles Figure 5-7. In this dialog box, click Fonts & Colors to obtain the Fonts & Colors dialog box shown in Figure 5-8.

Several options in the Fonts & Colors dialog box may need adjustment, depending on your system and your preferences:

- ✓ **Font selections**—You can set your serif, sans serif, and monospace fonts to whatever you desire using the selection boxes of the same names. If a Web page doesn't specify a font by name, Firefox will then use the font you tell it to use.
- ✓ **Default proportional font**—The Proportional field tells Firefox whether you want to use your serif or your sans serif font as the default font for pages that don't specify fonts.
- ✓ **Display resolution**—X tries to determine your screen's resolution in dots per inch; however, X sometimes gets this detail wrong. This isn't a major problem in most cases, but sometimes the result can be preposterously large or small fonts. Setting the resolution to a specific value in the Display Resolution field can help. (Divide your screen's horizontal resolution in pixels by its width in inches to determine your true resolution. You may need to round this value, since Firefox only presents a few options.)
- ✓ **Font sizes**—The Size (Pixels) fields provide a means to set the default font size in pixels for your proportional and monospace fonts.

**Figure 5-7**

The Firefox Preferences dialog box enables you to adjust many Firefox features.

- ✓ **Minimum font size**—The Minimum Font Size field lets you override font size settings in Web pages that display fonts that are too small for your taste.
- ✓ **Font override**—The Always Use My Fonts check box, if checked, forces Firefox to use the fonts you specify, rather than the ones encoded in Web pages. This can be handy if Web pages specify particularly ugly fonts. The Colors check box on the same line does the same for colors, which can be handy if Web pages use particularly ugly colors, as they sometimes do.

Setting Firefox fonts can be tedious because you may need to make changes, close the Fonts & Colors dialog box, close the Preferences dialog box, check the results, and then reopen the Preferences dialog box and the Fonts & Colors dialog box to fine-tune the results. You might need to dance to this tune several times before you get everything right. Another problem is that many Web pages specify their own fonts, so you might make changes and then discover that they don't have any effect because the page you're using as a reference uses its own fonts. (You can use my own Web page, <http://www.rodsbooks.com/books/> as a test page, if you like. It doesn't try to set any fonts.)

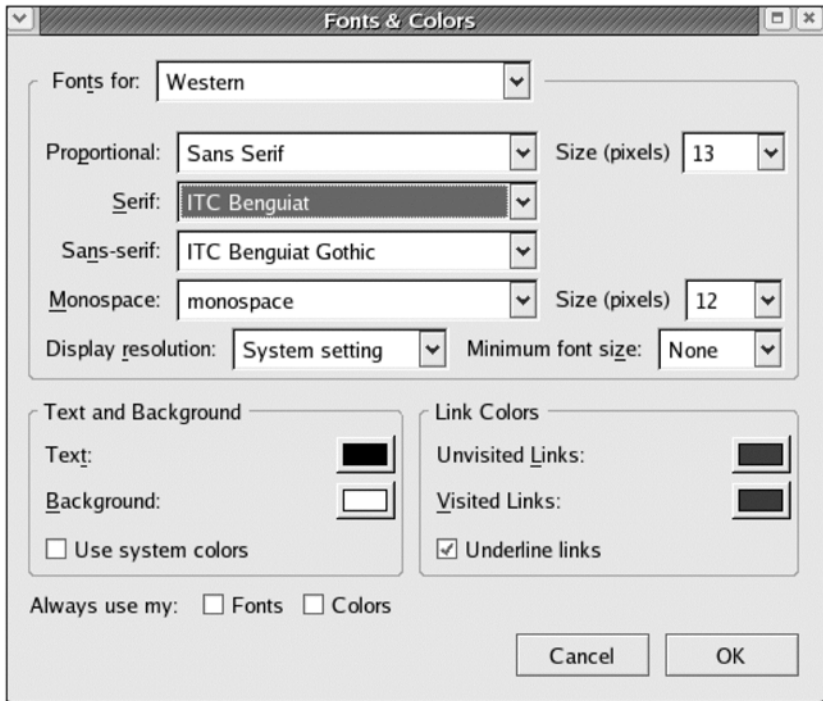


Figure 5-8

You can pick your preferred fonts in the Fonts & Colors dialog box.

Unfortunately, some Web pages are designed with a very narrow range of Web browsers in mind—typically, they work with Microsoft’s Internet Explorer. These pages can display poorly in Firefox or in other browsers. Other pages make assumptions about font size (in pixels), which can cause text to overlap with graphics or with other text. These problems can be real annoyances, but they’re usually the result of sloppy Web design. Sometimes you can work around such problems by adjusting your font size.

TIP: Firefox includes menu options—*View > Text Size > Increase* and *View > Text Size > Decrease*—that increase and decrease the size of text. These can be quick ways to temporarily adjust the text size if just one page is causing you problems.

Improving Firefox’s Security

Beyond fonts, the biggest source of gunk for any Web browser is security. The Web has become such a popular tool that all sorts of legal and illegal threats are lurking for the unwary. Knowing about these problems will help you avoid the pitfalls. These include threats from scripts that can be run on your system from Web pages, cookies, password management issues, and encryption issues.

NOTE: Some Web security issues can be addressed through the use of a proxy server. These tools are described in Chapter 10, "Network Degunking."

Java and JavaScript Issues

Sun's Java (<http://www.sun.com/software/learnabout/java/>) is a popular programming language. One of Java's big draws is that it provides explicitly cross-platform GUI tools. In most programming languages, creating a GUI application requires using either platform-specific tools or a cross-platform set of development libraries. In Java, that's all handled for the programmer; a call to display a dialog box will work in Windows, Linux, or Mac OS. Furthermore, Java is a scripting language, meaning that Java programs will run on any supported computer, without any need to recompile the software. (Recompiling can be a major hassle, particularly on cross-platform applications.)

JavaScript is similar to Java in that it's a cross-platform scripting tool. The two have a few surface similarities, but they aren't closely related, despite the similarity in their names.

In any event, both Java and JavaScript are appealing tools for Web designers who want to create complex interactive Web pages. For instance, suppose you want your Web page to compute a car's gas mileage if given miles driven and number of gallons consumed. You might write a Java or JavaScript applet to do the job and deliver it to the Web browser. Your site's users could then enter all the numbers they want and not consume your Web server's CPU time and network bandwidth doing this computation.

Unfortunately, these very features also make Java and JavaScript at least potentially threatening to users. Suppose you're a radical opponent of the word *grime*. You think it should be banned from the English language. You might set up a bogus Web site (even one with unrelated content) that delivers a Java or JavaScript applet that scours users' computers of the word *grime*. Any document that contains this word is deleted. This may be great for radical anti-*grime* crusaders, but for the rest of us, it's an invasion. This is a fairly simple (and silly) example, but you get the idea: By running Java or JavaScript programs via your Web browser, you're giving partial control of your computer to whoever happens to have written a Web page.

Both Java and JavaScript provide safeguards to prevent some of the worst possible abuses, and Linux's multiuser nature means that only so much damage can be done if you're sensible and don't run a Web browser as root. What's more, malicious Java and JavaScript code is likely (but not certain) to be targeted at Windows users, so you're safer in this respect if you're running Linux than if

you're running Windows. Still, Java and JavaScript do pose a danger. For the safest possible configuration, you should disable both. You can do this from the Web Features section of the main Firefox Preferences dialog box, as shown in Figure 5-9. Uncheck the Enable Java and Enable JavaScript options.

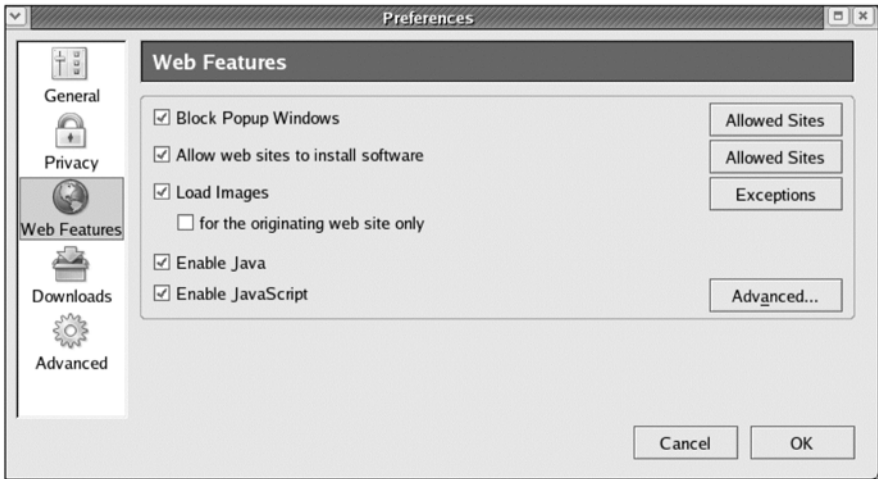


Figure 5-9

Most Web browsers enable you to disable Java and JavaScript for improved security.

NOTE: Web browsers support JavaScript natively, but to run Java applets, you must have a Java interpreter installed on your system. Thus, even with Enable Java checked, you might not be able to run Java applets. You shouldn't rely on this limitation if you intend to disable Java, though; you might forget about it and install a Java interpreter for reasons unrelated to your Web browser, at which point you'll become more vulnerable.

Unfortunately, a huge number of sites rely on JavaScript for proper functioning, and a smaller number rely on Java. Thus, disabling these features will shut you off from the normal operation of many Web pages.

TIP: You may want to configure Firefox (or whatever your preferred browser is) to refuse Java and JavaScript. You can then configure another browser to accept JavaScript and a third to accept Java and JavaScript. If you run into a page that isn't working correctly and you suspect it needs Java or JavaScript, you can then fire up the browser you've configured to use these features. You can perform a similar trick with a single browser by employing multiple Linux accounts. Use your regular account for a safe configuration and then set up extra accounts in which you configure your browser to enable these features. You can even run programs from multiple accounts on the same display by using `su` to change your login name, as in typing `su java` in an xterm window to become the `java` user and run a browser as that user.

Although not technically Java or JavaScript issues, you can use the Web Features configuration area to block other types of actions too, such as pop-up windows. These windows are often used by advertisers and are much hated by most users, so blocking them can make sense. Clicking the Allowed Sites button to the right of the Block Popup Windows line enables you to specify exceptions. This may be necessary to enable sites with legitimate pop-up windows to work as designed.

Cookies

Baked sugar, flour, eggs, and other ingredients can certainly create gunk in the real world. That gunk can be cleaned up with a dishwasher, a sponge, or other real-world cleaners. Web-based cookies are another matter, though. The idea behind these virtual confections is to give Web sites a way to track users for the benefit of features like e-commerce sites' "shopping carts." When you add an item to your shopping cart, the Web server delivers a cookie to your computer. When you finish shopping, the Web server asks for the cookie and your computer returns it, enabling the Web server to ring up the correct purchase. Cookies can also be used in longer-term transactions—say, to enable a Web site to remember your preferences for fonts or the amount of detail you want to see.

Unfortunately, cookies also have another possible use: to track your online activities. Many people object to such tracking, viewing it as an invasion of privacy. Some of the worst problems are associated with online advertisers. A handful of companies dominate the online advertising that you see and deliver the ads directly from their own server rather than from the servers of the sites you think you're visiting. This fact enables advertising agencies to track the sites you visit, even if the sites themselves are unrelated, and even if they display different ads—if the ads are fed through the same ad agency, they can use cookies to track you. This ability gives them data on correlations between sites—say that visitors to a baby care site also visited an automotive site to view information on minivans. If a visitor then goes to a hunting site, followed by a porn site, that information can also be added to the database. The result can be an unprecedented capacity for online advertisers to track you as an individual.

If this prospect disturbs you, you may want to look into Firefox's cookie management tools, which are accessible from the Privacy area of the Preferences dialog box, as shown in Figure 5-10. The Allow Sites to Set Cookies option enables cookie features as a group. Checking For the Originating Web Site Only restricts the cookies that Firefox accepts to the main Web site. Using this option can help keep advertisers' cookies off of your computer, but it's not a foolproof option, and it can also have unintended consequences. You can also view the cookies that are currently set by clicking View Cookies. The result is

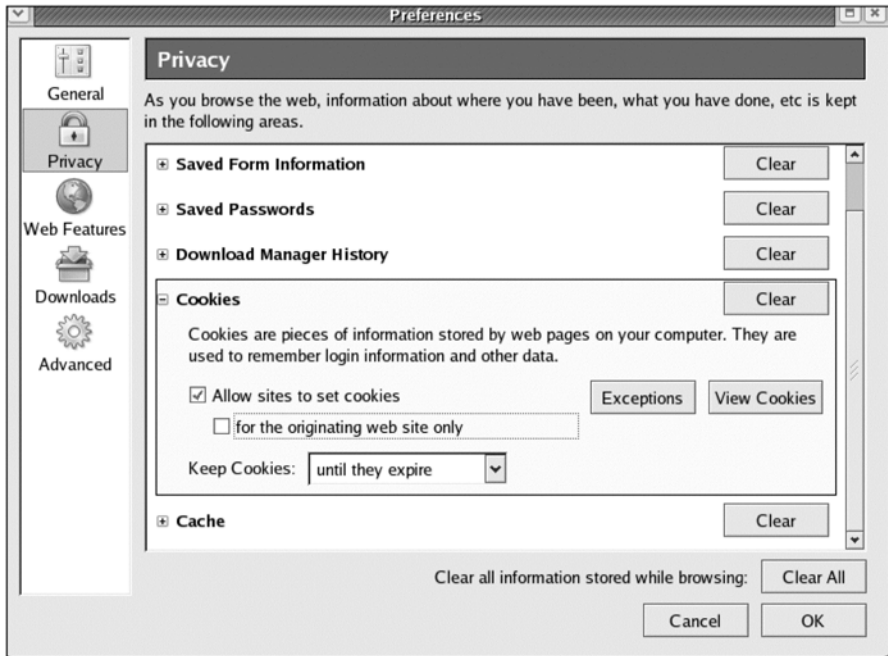


Figure 5-10

The Privacy section of Firefox's Preferences dialog box lets you control cookies and other potential privacy threats on the Web.

a list of cookies, most of which look like gibberish. You may want to go through them and delete those from sites you don't recognize—many of them are potential privacy-invasion tools.

Password Management

Increasing numbers of Web sites demand passwords. Remembering all of these passwords can be a major undertaking. You might prefer to spend your time memorizing Homer's *Odyssey*—at least it's interesting, unlike a list of passwords. Alternatively, you might employ just one username and password on all your Web sites. This approach has its problems, too—if miscreants obtain that one username and password, they can wreak a lot of havoc. For this reason, at a bare minimum, you should use unique passwords for particularly sensitive sites, such as online banking sites, and leave your reused passwords for relatively uncritical sites, such as online newspapers that require registration before they'll let you read their content.

Firefox, like all major modern Web browsers, provides a password management tool. Firefox can usually correctly identify a page that asks for a password and

will ask you if you want Firefox to remember the password for you. If you respond in the affirmative, the program stores your password in a file on your hard disk. The next time you visit the site, Firefox recognizes that fact and fills in the username and password, thus easing your memory load and enabling you to use unique (and random) passwords for every site you use.

NOTE: Chapter 9, “Account Degunking,” includes a section that describes how to construct a good password. Although the details of password length, types of characters, and case sensitivity vary between sites, the same principles that make a good Linux password also make a good password for a Web site or a non-Linux system.

The Privacy area of the Firefox Preferences dialog box provides a way to manage these passwords. Click Saved Passwords and you can tell Firefox whether to prompt you to save passwords. You can also review your existing passwords, delete the password for a site, and review the list of sites for which you don’t want Firefox to manage the passwords.

Like so many computer issues, though, having Firefox manage your passwords isn’t without its drawbacks. One of these is that the password file is vulnerable. An intruder could snatch it and gain access to all your sites. For this reason, I recommend *not* letting Firefox manage your most sensitive passwords, such as those for online banking services. Another problem is that you might lose all your passwords in a disk crash or to an accidental deletion of your Firefox dot files. If you change browsers, you’ll have to re-enter all of your passwords, which can be a hassle at best.

SSL Encryption

You should be very aware of the encrypted or nonencrypted nature of your network traffic. Most Web sites aren’t encrypted, so data sent to you or returned by you in forms or URLs can be read on intervening computers, or sometimes by other computers on your local network, or on the server’s local network. This fact is a major problem when transferring sensitive data, such as credit card numbers.

WARNING! Many sites that request passwords don’t bother to encrypt the password data. This is one of the reasons you should use unique passwords for your most sensitive Web sites.

Fortunately, most sites that transfer sensitive data do so using the Secure Sockets Layer (SSL) encryption protocol. SSL provides a way to encrypt network protocols, meaning that the data stream is scrambled in such a way that anybody monitoring it will see nothing but gibberish.

Most Web sites that employ encryption do so over the Secure HTTP port, which is identified in browsers by a leading **https://** rather than **http://** in the URL. This indication isn't 100-percent proof of SSL use, though; a poorly configured Web browser (or worse, one that's been configured dishonestly) could use the usual Secure HTTP port but not use encryption. Thus, you should always look for an indicator of the encryption status in the Web browser's window. In most Web browsers, including Firefox, this is denoted by a key or closed padlock icon in the bottom-right or bottom-left corner of the window. (Even within Firefox, the precise icon used and its placement varies from one version to another.) Firefox also changes its URL entry field color to gold for secure sites, but most browsers don't do this. Look for a secure connection indicator before you send a critical password, credit card number, or other sensitive information to a Web site.

Alternatives to Firefox

Although Firefox has received a lot of attention prior to and immediately after the release of its first version (that is, Firefox 1.0), it's far from the only game in town. There are other Web browsers for Linux:

- ✓ **Mozilla**—This browser is Firefox's big brother—or if you prefer, Firefox's overweight sibling. Mozilla incorporates lots of extra features, some of which are being spun off into separate programs. For instance, the full version of Mozilla includes a mail reader, but that's being spun off into a separate program (Thunderbird), which you can use or not use with Firefox, as you see fit.
- ✓ **Netscape**—Netscape is Mozilla's commercial twin. Although a Linux version is available, it doesn't ship with most Linux distributions and is seldom used. The open source Mozilla is more convenient because it ships with most distributions.
- ✓ **Galeon**—This browser is related to Netscape, Mozilla, and Firefox in that it uses the same rendering engine—that is, the same software to convert HTML to formatted text in the window. It's lighter than Mozilla and Netscape but isn't quite as polished as Firefox. If you like Firefox, it might be worth using as a second browser (say, one with Java and JavaScript enabled, as described earlier). Check <http://galeon.sourceforge.net> for more information.
- ✓ **Konqueror**—This program serves as both a file browser and a Web browser for KDE. Chapter 4, “Degunking Your Desktop Environment,” describes a few of Konqueror's file manager features. As a Web browser, it's similar in scope and capabilities to Firefox. Konqueror's rendering engine has also been used by Apple in its Safari Web browser, so if you hear that a

site works with Safari but not with Firefox, you might want to give Konqueror a try.

- ✓ **Opera**—This commercial browser, headquartered at <http://www.opera.com>, was once viewed as the lightweight alternative to Mozilla or (on Windows) Internet Explorer. Firefox has largely stolen that bit of thunder, but Opera remains a capable browser. You can either buy a copy outright or live with the free version, which displays ads in its menu bar.
- ✓ **Lynx**—This browser is unusual because it's text based. You can run Lynx from a text-mode login or within a single xterm window from a GUI login. Of course, being text based, Lynx doesn't display graphics, but this fact can be an advantage as well as a drawback—graphics files are typically much larger than the main Web page files, so Lynx is much faster than most browsers. Lynx ships with most Linux distributions, and you can learn more at <http://lynx.browser.org>.

Notable by its absence from this list is Microsoft's Internet Explorer. Although this browser is the most popular one on the Internet, it's not available for Linux. You can run Internet Explorer using WINE Is Not an Emulator (WINE; <http://www.winehq.org>), a program for running Windows programs under Linux. Unless you need to access a site that simply will not work with other browsers, though, I don't recommend doing this. Internet Explorer has developed quite a reputation for having serious security flaws. Although many of these flaws aren't as serious when running under Linux and WINE as under Windows, there's no point to opening yourself up to the added risk. Furthermore, configuring and running WINE can be tricky. There's no point in going to the effort just to run Internet Explorer unless you really must access a site that simply won't work with anything else.

Degunking Evolution

The Linux office suite and Web browser arenas are dominated by just one program each. (In the case of Web browsers, that domination shifted rather rapidly from Mozilla to Mozilla Firefox during 2004.) This is less true of the e-mail client realm; Linux supports many e-mail clients, and no one really dominates the field. One of the largest e-mail readers is also one of the most popular, though, and so I describe it here. This e-mail client is Novell Evolution (<http://www.novell.com/products/desktop/features/evolution.html>). Other Linux mail clients can be degunked in a similar fashion, although some details differ.

Specific areas that you can degunk include reducing clutter (both in the options the program presents to you and in keeping your mail organized), setting mail reader HTML and security options, and investigating the possibility of using a simpler mail client.

Reducing Clutter in Evolution

E-mail client clutter comes in two forms: too many options presented by the program itself and too many mail messages staring back at you from your inbox. Both types of clutter can be tackled, but they require different approaches.

Configuring Evolution to Reduce Clutter

Evolution aims to be a very full-featured mail client. It also functions as a contact manager, calendar, and task manager. Thus, Evolution's default main window presents a wide array of buttons and options, as shown in Figure 5-11. Features include a GUI menu bar at the top of the window, an icon toolbar for common functions just below that, a tool for searching your mail, a list of mail

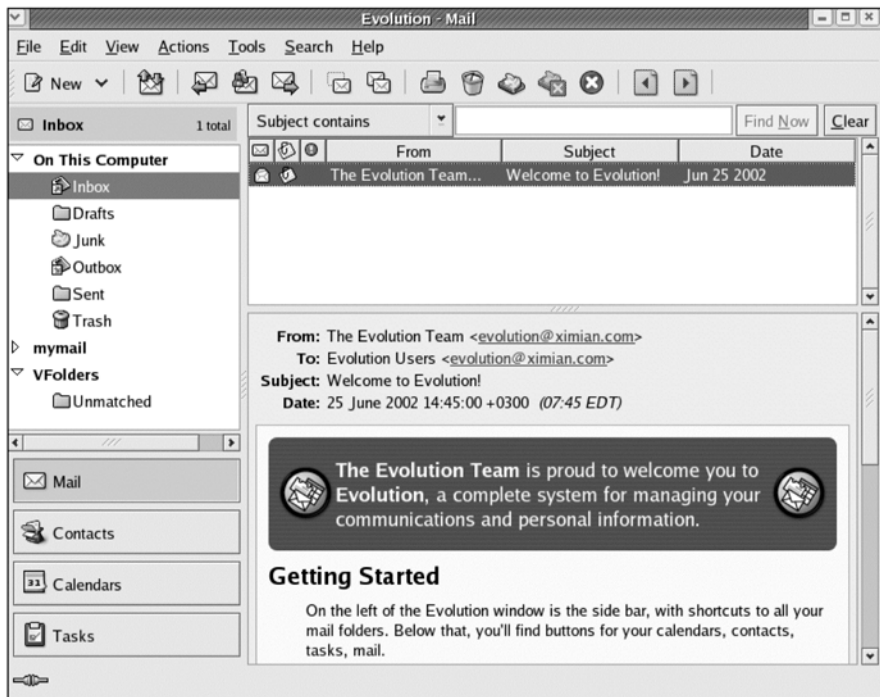


Figure 5-11

Evolution is a full-featured mail reader with lots of options, many of which manifest themselves in the program's main window.

folders, a list of messages in the folder you've selected, a preview pane showing the text of the message you've selected, and buttons to select the major Evolution mode (mail, contact manager, and so on). Although these tools are all useful, you can simplify the Evolution configuration if you seldom use some of these features, thus providing more window real estate for reading messages or scrolling through message lists.

Specific features you can adjust include the following:

- ✓ **Toolbar**—Not everybody wants or needs the icon toolbar. To toggle it off or on, select **View > Toolbar**. You can also move it around the window or even out of the main Evolution window by clicking on its far left edge and dragging it around. Evolution remembers whether to display the toolbar when you restart it, but if you move the toolbar to a new location, you'll need to move it again when you restart the program.
- ✓ **Message preview pane**—You can toggle the message preview pane by selecting **View > Preview Pane**. If you eliminate the preview pane, you must double-click a message to read it. When you do this, a new window opens with the text of the message.
- ✓ **Message summary information**—Evolution provides highly customizable summary information in the message list pane. You can sort by any field, add fields, delete fields, define named sets of data you want displayed, and switch between these sets. To do so, use the **Define Views** tool: Click **View > Current View > Define Views**. The result is a dialog box in which you can select a named view and edit it. You can add or delete fields, change sorting options, and so on. These tools can be very handy in helping you organize your messages, particularly if you need to archive a lot of them.
- ✓ **Headers**—E-mail messages arrive with a set of *headers*, which are lines describing the message's source, destination, computers that have relayed the message, the name of the originating mail client, and so on. By default, Evolution presents just a few headers (**From:**, **To:**, **Subject:**, and **Date:**), as shown in Figure 5-11. You can configure it to display all the headers by selecting **View > Message Display > Show Full Headers**. The **View > Message Display > Show Email Source** option also shows all the headers, but it also causes Evolution to ignore HTML formatting. Displaying headers in this way is mainly useful in tracking down e-mail delivery problems or in learning the source of spam.
- ✓ **Fonts**—You can temporarily increase or decrease the size of text shown by using the **View > Text Size > Larger** and **View > Text Size > Smaller** options. To permanently change the text size or use a different font, select **Tools > Settings**. This brings up the Evolution Settings dialog box. In the **Mail Preferences** area, you can set the font that Evolution uses for message display. (It uses options inherited from GNOME by default.)

TIP: If you make changes to these options but find that you want a still less-cluttered display, you may need to use another mail client, as described shortly, in "Alternatives to Evolution." Evolution provides a certain degree of configurability, but some GUI controls, such as the buttons to switch between its major function types, simply cannot be disabled.

Organizing Your Mail

E-mail can seem like a never-ending flood. There's e-mail from friends, e-mail from relatives, e-mail from co-workers, e-mail from Web retailers, and don't even get me started on spam! (Seriously, spam and e-mail worms and viruses are covered in Chapter 10.) Organizing this flood of e-mail can be a serious challenge. One of the first keys to doing so, though, is to organize the messages into folders.

To create a folder, right-click the On This Computer line in the folder list on the left side of the Evolution window in Figure 5-11 and pick New Folder from the resulting pop-up menu. You can then create a folder that will hold messages. You can create folders to hold whatever types of messages you like. You might organize folders by sender, by work project, by mail type, and so on. The important thing, as with organizing files on your hard disk, is to use the folders effectively. Don't let messages languish in your inbox for too long, lest they end up forgotten. File your important messages and discard those you don't want to keep.

NOTE: If you use the Post Office Protocol (POP) to retrieve mail or if your Linux system functions as its own mail server, you'll organize your folders locally. In Evolution, this is done in subfolders of On This Computer. If you use the Internet Mail Access Protocol (IMAP) to retrieve mail, though, you have the option of storing mail in folders on the mail server computer. Doing so enables you to access your messages from multiple computers (say, a desktop system and a laptop system). Some ISPs delete messages from their servers after a while, though, so check your ISP's mail retention policies before relying on folders stored on the mail server. You can create folders on an IMAP server much as you do on the local system. In Figure 5-11, the mymail entry in the folder list refers to an IMAP server. (The name is likely to be different for your system.) Right-click it and pick New Folder, just as you would to create a local mail folder.

GunkBuster's Notebook: Setting HTML and Security Options

E-mail worms and viruses are a serious threat to Windows systems. Sometimes it seems that every week there's a new Windows e-mail worm threatening to take down the Internet like some crazed monster from a 1950s B movie. Fortunately, Linux has been relatively untouched by such problems, for three reasons. First, Linux doesn't yet have the market share on the desktop to be a viable target for the miscreants who write computer worms and

viruses. Second, no one Linux e-mail program dominates the Linux e-mail client market, which makes the task of writing an effective Linux e-mail worm rather difficult. Third, Linux e-mail programs tend to be written with more secure default settings than certain popular Windows e-mail programs. For these reasons, the main threats to Linux systems from e-mail worms is the nuisance factor of having to sift through and dispose of the worms that are written for Windows. Chapter 10 covers this topic, describing some approaches you can take to help automate this odious task.

Nonetheless, you shouldn't take e-mail security lightly. It wasn't so long ago that e-mail worms on Windows were purely works of fiction; it's conceivable that some clever miscreant will discover a way to work around the problems associated with writing a Linux e-mail worm. As protection against such a possibility, you should review your mail client's configuration to look for possible security holes. In Evolution, you can do this from the Evolution Settings dialog box, which you can reach by selecting Tools > Settings. Click the Mail Preferences option and select the HTML Mail tab. The display should resemble Figure 5-12.

For the absolute best security, you should completely disable HTML options. Although HTML enables senders to specify fonts, use italic text, and even embed images in e-mail, it can be a security nightmare. HTML references to images on the Web enable the sender to track whether and when you read the message, install cookies in your Web browser, and conceivably even run Java or JavaScript programs on your system, much as Web pages can do.

Unfortunately, Evolution's only option to completely disable HTML is the View > Message Display > Show Email Source item mentioned earlier with respect to message headers. This is an awkward way to dispose of HTML because you must then look at all the headers, even when you don't want to. Evolution does, though, provide options to restrict what HTML can do. The default configuration is to never load HTML-specified images from the Web, for instance, which is sensible. You can enable automatic loading of images if you like, but I advise against it.

Most e-mail clients offer the option of *sending* HTML mail. Enabling this option lets you use italics, large fonts, colored text, and so on. As I've said, though, the safest configuration for recipients is to ignore all HTML formatting; thus, your

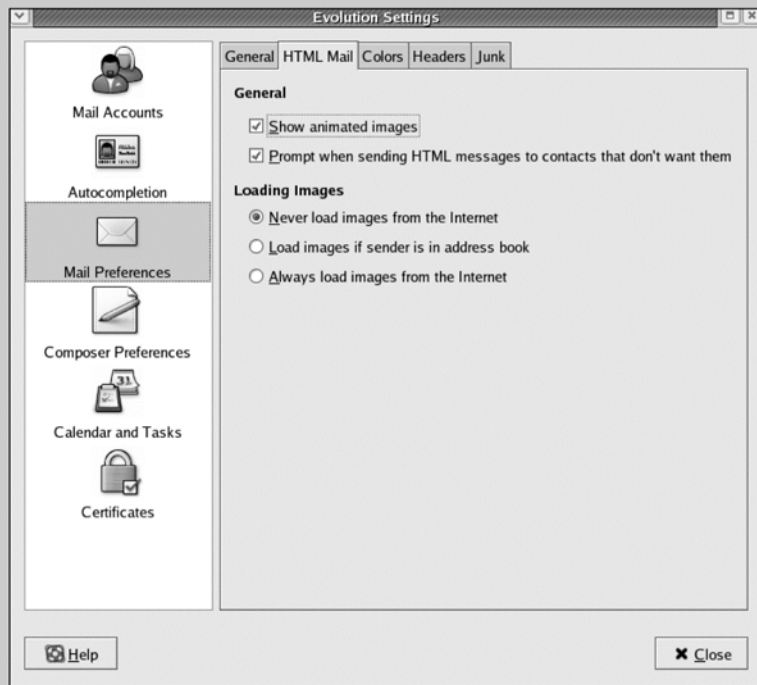


Figure 5-12

Many e-mail security issues are related to HTML.

correspondents might never see such formatting. In fact, some people get annoyed when they receive HTML messages—older mail clients might not know how to parse them, the result being that the readers will end up seeing “raw” HTML, which is harder to read than plain text. Thus, it’s probably best to send non-HTML text at all times. You can usually control this option from somewhere in the mail client’s configuration tool. In the case of Evolution, the sending of HTML can be enabled or disabled from the Composer Preferences area. Click that option and be sure the Format Messages in HTML option is unchecked (or checked, if you really want to send HTML e-mail). A related option is in the Mail Preferences area, on the HTML Mail tab shown in Figure 5-12. The Prompt When Sending HTML Messages to Contacts That Don’t Want Them option coordinates with the Evolution contact manager. Each contact manager entry has a box you can check if the individual wants HTML messages. If this box is unchecked and you’ve set the appropriate options in Evolution’s mail configuration, the program will prompt you before sending HTML-formatted mail to the person.

Alternatives to Evolution

As with other major applications, alternatives exist for Linux e-mail clients. Evolution is one of the most feature laden of these programs, but as you've read repeatedly in this book by now, the program with the most features isn't necessarily the best one. You might want something slimmer, something that integrates better with a specific desktop environment, something with specific features that aren't in Evolution, or something that's otherwise different. Some of the more popular alternatives include the following:

- ✓ **Thunderbird**—This program is the e-mail companion to the Firefox Web browser. It was specifically designed with security in mind, although that emphasis is more of an issue in the Windows world than in the Linux world, since Linux mail clients as a group are fairly safe. You can learn more at <http://www.mozilla.org/products/thunderbird/>.
- ✓ **KMail**—This program comes with KDE (check <http://kmail.kde.org> for details). It's lighter than Evolution but still provides the features that users expect in a modern e-mail client.
- ✓ **Balsa**—Although Evolution is loosely associated with GNOME, another GNOME-related mail client is Balsa (<http://balsa.gnome.org>). Balsa doesn't get a lot of publicity, but it's a capable mail client, particularly for those who like GNOME but want something slimmer than Evolution.
- ✓ **Sylpheed**—This program, <http://sylpheed.good-day.net>, is another mail client with loose ties to GNOME. It's similar in overall capabilities to Balsa, but of course the two programs differ in several details.
- ✓ **Mutt**—Mutt (<http://www.mutt.org>) may be the most popular text-based mail reader. This program offers a plethora of features but can be used from a text-mode login or within an xterm window.
- ✓ **Emacs**—Over the years, the Emacs text editor (<http://www.gnu.org/software/emacs/emacs.html>) has accumulated an almost frightening number of features. Among these is the ability to handle e-mail. If you already use Emacs for lots of other things, you might want to look into its e-mail capabilities. Emacs is primarily a text-mode program, although it does enable some GUI extensions when run from X.

This list only begins to scratch the surface. Linux e-mail clients are particularly numerous. You can try doing a Web search or checking a site such as Tucows (http://linux.tucows.com/email_clients_default.html) for pointers to more Linux e-mail clients if you're picky and find that none of these programs suits your taste.



Cleaning Out Unused Packages

Degunking Checklist:

- ✓ Understand why removing unused packages is necessary.
- ✓ Determine what packages are installed on your system.
- ✓ Determine what installed packages are unused.
- ✓ Remove unused packages.
- ✓ Keep the packages you retain up-to-date.
- ✓ Understand package dependencies and deal with problems arising from them.

All major Linux distributions deliver software in *packages*—collections of files that make up a working program or other logical set of files, such as a set of related fonts. Typically, a program package includes one or more executable program files, documentation files, configuration files, and possibly support libraries. A Linux package is usually distributed as a single file that contains all these files, much like a Zip archive file contains many files. Package files, though, include extra information on *dependencies* (the additional packages upon which the first package relies), package descriptions, and so on. Most Linux distributions use the RPM Package Manager (RPM) tool for package management, but some use other systems. For instance, Debian and its derivatives use Debian packages, Slackware uses tarballs, and Gentoo uses a unique source-based system.

A working Linux system consists of hundreds of installed packages. You can add or remove packages to install or uninstall programs, within limits imposed by package dependencies, free disk space, and so on. The package systems used by Linux distributions are very useful, but the ease of installing packages means that you can quickly collect quite a lot of unnecessary packages. These packages consume disk space and can cause additional problems, as described shortly. Thus, you should periodically review your system's configuration to weed out unused packages. This chapter will help you do this. I'll show you how to use package browsers to help identify installed packages, how to determine which packages are unnecessary, and how to remove these unnecessary packages. I'll also show you how to keep your system up-to-date—old packages can contain bugs (some of which can cause security problems), so replacing old software with new ones is very useful. Finally, I'll help you degunk dependency problems—problems that can arise because of conflicting dependencies.

Why Clean Out Unused Packages?

In today's world of hard disks with sizes measured in the hundreds of gigabytes, you might wonder why cleaning out unused packages is necessary. The answer is multifaceted. Unused packages can consume disk space, cause administration problems, pose security risks, and complicate subsequent system or package upgrades. Each of these issues could be critically important on some systems, but even if one or two don't concern you, chances are at least one should. Thus, I'll describe each of these problems in a bit more detail so you can understand the consequences of package gunk.

Reducing Disk Space Consumption

The most obvious reason to remove unused packages is to minimize disk space consumption. This reason may not seem important if you've got hundreds of gigabytes of unused space, but if your system has little disk space, it's an important issue. Furthermore, sometimes having a lot of free space can be important—say, to give you space to create and manipulate a large temporary file, such as a large multimedia file.

You should also note that most Linux systems consist of multiple partitions. Most Linux packages install primarily to the `/usr` directory, which often resides on its own partition. If not, most packages install in the root (`/`) partition. Even if you've got plenty of free space in some other partition, such as the partition for `/home` (which holds user files), you might not have much free space where most packages reside. You can use the `df` command to determine how much free space you've got on various partitions, as described in Chapter 3, “Degunking User Files.”

Maintaining Administration Simplicity

One of the problems with Linux, particularly for new users, is that it is a very complex operating system. You've got a boot loader, the kernel, system startup scripts, daemons, cron jobs, user accounts, and lots more. The result of all this complexity is increased administrative burden. You must manage all these systems, and if you make a mistake, the computer can behave strangely or even do things that are risky. Thus, minimizing the complexity of a Linux system is desirable. Desktop-oriented distributions try to hide as much of this complexity as possible, but they can only go so far in this respect.

One way to help minimize the complexity of a Linux system is to reduce the number of installed packages—that is, to uninstall the ones that aren't necessary. If the Apache Web server, to name just one example, isn't installed, it needn't be configured or administered in any way.

Of course, you shouldn't take this too far. A Linux system with no packages installed isn't a Linux system at all—it's a blank hard disk! Nonetheless, removing those packages that you really don't need from the system can be a good way to help reduce the administrative burden of running the Linux system.

Reducing Security Risks

Whether it runs Linux, FreeBSD, Solaris, Mac OS, Windows, or any other OS, one of the major problems with any computer is security. Some computer security problems relate to the computer's physical environment (locks on doors, for instance) or the computer's users (they might write down passwords, for instance). Many computer security problems, though, arise because of software bugs.

Some security bugs are associated with servers, such as the Apache Web server or the sendmail mail server. Servers are essentially programs that users on other computers can run. Thus, bugs in such programs can sometimes enable outsiders to gain control of your computer even if they don't have local accounts.

Other security bugs relate to local programs—that is, programs that only local users can run. Usually, bugs in such programs aren't really security issues, although they can be annoying. But sometimes, such bugs can enable local users to gain root access to the computer. Such bugs are most important on systems that are used by many people, but they can be exploited by outsiders. For instance, if a miscreant gains access to your ordinary user account via a remote login protocol (say, because your password was intercepted), the miscreant might then be able to abuse a local program bug to gain root privileges.

In both cases, security bugs can be dealt with in several ways, including keeping your system up-to-date, as described later, in “Keep Software Up-to-Date.” One of the best ways to minimize the risk, though, is to reduce the amount of unnecessary software installed on the computer. If a bug in SuperProg can be abused to gain root access, for example, and if you don't need SuperProg, keeping it off the computer can guard against abuses. Of course, you can't know which programs are subject to abuse—vulnerabilities lurk, undiscovered, in many programs. As these problems are found, updates are issued, but if you keep unnecessary programs off of your computer, you won't need to worry about updating them and your system won't be subject to attack in the interval between the vulnerability being discovered and its being fixed.

Minimizing Upgrade Headaches

A final reason for removing unused packages is to reduce upgrade complications. Earlier, I noted that Linux packages include dependency lists. As packages are updated, these dependencies may change. For instance, SuperProg 1.3 may depend on HyperLib 2.8, but SuperProg 1.4 might depend on HyperLib 2.9. Thus, if SuperProg exists on your system and you use a tool to help automate the update process, you'll end up upgrading both SuperProg and HyperLib.

Unfortunately, such upgrades sometimes cause problems. For instance, HyperLib might be used by some other program (say, OtherProg). Perhaps OtherProg doesn't need updating when SuperProg is updated, but the new version of HyperLib might not be compatible with your version of OtherProg. Thus, the update of HyperLib that's mandated by the upgrade of SuperProg will cause problems for OtherProg. In the best of all possible worlds, your package management system will spot this problem and alert you to it, so you'll be able to intervene in the upgrade process before harm is done. (The upcoming section, "Handle Dependency Problems," describes some of the issues and solutions in more detail.) Unfortunately, problems sometimes sneak past the package management systems.

In practice, the symptoms that result from such problems can be quite varied. They range from a package system refusing to upgrade or install a program to once-working programs randomly crashing or otherwise misbehaving.

One way to reduce the risk of such problems occurring is to simplify your system by—you guessed it—removing unnecessary packages. To use the fictitious programs described earlier, if you hadn't installed SuperProg, an automatic system update wouldn't try to upgrade it. Depending on the package management system and how you initiated the upgrade, HyperLib might not be upgraded, so problems with OtherProg might not appear. (On the other hand, HyperLib might be upgraded despite the absence of SuperProg, but such an upgrade is less likely without SuperProg.)

Use Package Management Tools

Removing extraneous package gunk requires using your Linux distribution's package management tools. These tools differ from one distribution to another, but the two most common systems are RPM and Debian packages. I'll briefly describe both systems, including their command-line tools. For removing unnecessary packages, though, the best tool is arguably a GUI package browser. Several such package browsers exist, so I'll describe several and give examples of how to use a couple of them.

Understanding Linux Package Systems

In the Linux world, several package managers have emerged to help place files on the computer and manage them. The most popular package management system in Linux is RPM. This system is used by many distributions, including Fedora, Lycoris, Mandrake, Red Hat, SuSE, TurboLinux, and Yellow Dog. The second most popular Linux package management system is the Debian package

manager, which is used by Debian and its derivatives, including Libranet, Ubuntu, and Xandros. These package systems are incompatible with one another—you can't install an RPM using the Debian package tools or vice versa. (Some tools exist to convert between formats, though, and efforts are underway to better unify the two systems.) Despite their format incompatibility, the RPM and Debian package systems are similar in overall features and capabilities.

NOTE: *A few distributions, such as Gentoo and Stampede, use their own package management tools, which I don't describe here. Slackware Linux uses tarballs with a few extra features as a package file format. Again, I don't describe that system here. If your distribution doesn't use RPMs or Debian packages, you'll have to consult its documentation to learn how to manipulate packages.*

Using RPM Tools

The basic command-line tool for manipulating RPM packages is `rpm`. This program takes various options to tell it what to do. Typically, the result looks something like this:

```
# rpm -e superprog
```

In this example, `-e` is an option that tells `rpm` to erase (uninstall) a package, which is then named (`superprog`). Other options include `-i` (to install a package), `-U` (to upgrade a package), `-V` (to verify a package's correct installation), and `-q` (to query information about a package). You can add extra options to most of these to modify their behavior, as in `-Uvh` to upgrade a package and display a progress bar made up of hash marks (`#`).

Depending on the options passed to `rpm`, the program usually expects to see either a package name (such as `superprog` or `superprog-1.4-2`; the version number isn't always used) or a package filename (such as `superprog-1.4-2.i386.rpm`). Specifically, the `-e`, `-q`, and `-V` options take a package name, but the `-i` and `-U` options take a package filename. (You can query an uninstalled package by using the `-qf` option, which then takes a package filename rather than a package name.)

To clean out unnecessary packages, you'll make heavy use of the `-e` option. You might also use the `-q` option to query a package to learn what it does before deleting it. In particular, the `-qi` option displays information on the package:

```
$ rpm -qi coreutils
```

```

Name       : coreutils      Relocations: (not relocatable)
Version    : 5.2.1          Vendor: Red Hat, Inc.
Release    : 31             Build Date: Tue 05 Oct 2004 11:50:21 AM EDT
Install Date: Sat 04 Dec 2004 01:52:37 PM EST   Build Host:
                                                tweety.build.redhat.com
Group      :System Environment/Base Source RPM: coreutils-5.2.1-
                                                31.src.rpm
Size       : 7307074         License: GPL
Signature  : DSA/SHA1, Wed 20 Oct 2004 02:46:04 PM EDT, Key ID
              b44269d04f2a6fd2
Packager   : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL        : ftp://alpha.gnu.org/gnu/coreutils/
Summary    : The GNU core utilities: a set of tools commonly used in
              shell scripts

```

Description :

These are the GNU core utilities. This package is the combination of the old GNU fileutils, sh-utils, and textutils packages.

For the most part, this information is clearly labeled. If you don't understand a label, you can safely ignore it. Of most use in determining whether a package is gunk is the description, which appears at the bottom of the output. This example shows the core utilities for Fedora 3; this is a package that is *not* gunk!

Using RPM to query a package requires that you know the package name, but what if you don't know the package name? If you know a file that's part of the package, you can use `-qf` with the name of the file to learn the name of the package to which it belongs:

```

$ rpm -qf /bin/cat
coreutils-5.2.1-31

```

These tools can be very useful for identifying specific programs, but if you want to track down all the gunky unnecessary packages on your system, they aren't really well suited. For that task, a GUI package browser (described shortly) is a better choice.

Using Debian Package Tools

Debian's package tools and the RPM utilities work very similarly, but the details differ. The Debian tool that's most similar to `rpm` is `dpkg`. Like `rpm`, `dpkg` takes one or more options to tell it what to do. The `-i` option installs a package, `-r` removes a package but leaves configuration files intact, `-P` purges the package

(that is, it removes everything, including configuration files), `-p` displays information about an installed package, and `-S` searches for the package that owns a file. For instance, suppose you want to remove the `superprog` package. Either of the following commands will do so:

```
# dpkg -r superprog
# dpkg -P superprog
```

The first of these commands won't remove configuration files, if the package has any. (Many packages lack configuration files, so these two commands may have identical effects.) These two commands are very handy in cleaning out gunky packages, but you might want to use `-S` to locate the package associated with a file:

```
$ dpkg -S /bin/cat
textutils: /bin/cat
```

NOTE: *Debian places `/bin/cat` in the `textutils` package, whereas Fedora places the same file in the `coreutils` package, as shown earlier in "Using RPM Tools." Such differences are common between distributions. Even two distributions that use the same package system aren't guaranteed to place the same file in the same package.*

You can use the `-p` option to `dpkg` to display information on an installed package. As with `rpm`, this summary includes assorted details about the package, including a description of its purpose:

```
$ dpkg -p textutils
Package: textutils
Essential: yes
Priority: required
Section: base
Installed-Size: 1728
Maintainer: Herbert Xu <herbert@debian.org>
Architecture: powerpc
Version: 2.0-12
Replaces: bsdmainutils (<= 4.5.2), ptx
Provides: ptx
```

```

Pre-Depends: libc6 (>= 2.2.4-4)
Conflicts: ptx
Filename: pool/main/t/textutils/textutils_2.0-12_powerpc.deb
Size: 567046
MD5sum: 9bf76cdd00a0f7a3dd120cf99de4fed6
Description: The GNU text file processing utilities.
The utilities: cat cksum comm csplit cut expand fmt fold head join
md5sum
nl od paste pr ptx sort split sum tac tail tr tsort unexpand uniq wc.

```

One feature of this output that's not present in the rpm equivalent is the `Priority:` line, which gives a clue to a package's importance. In this example, this line specifies `required`, meaning that the package should never be uninstalled.

In addition to `dpkg`, Debian supports a variety of higher-level tools, known as the Advanced Packaging Tool (APT). This suite, and particularly a command called `apt-get`, enables you to issue a single command to have the system retrieve a package from your installation media or an Internet site and install it:

```
# apt-get install superprog
```

This command retrieves the `superprog` package from your installation media (if it's present there) or from an Internet site (if you've so configured APT) and then installs the package on the system. To work over the Internet, APT requires that you tell it about APT Internet archives in the `/etc/apt/sources.list` file. The default file provides some sample entries that have been commented out with hash marks. Debian's APT HOWTO document (<http://www.debian.org/doc/manuals/apt-howto/>) provides more information on this task. Clearly, APT is a handy tool for installing packages, but this convenience comes at the cost of making it easy to install a lot of gunk. Furthermore, APT keeps copies of the original package files it installs on the hard disk, which is itself a type of gunk. Typing `apt-get clean` will clean out this package cache, so doing this after you install packages using `apt-get` can help reduce clutter.

As with RPM-based systems, the easiest way to locate gunky packages on Debian-based systems is to use a GUI package browser, as described next.

Options for Package Browsers

A package browser is a tool for perusing information about installed (and sometimes uninstalled) packages. Using a package browser, you can go through all the packages installed on your system, read their descriptions, and uninstall packages that seem gunky.

Unfortunately, package browsers aren't quite as well standardized as the package tools upon which they rely. Many distributions provide their own GUI tools for package management. These include Debian's text-mode `dselect` and X-based Synaptic, Fedora and Red Hat's Package Management, and SuSE's text-mode YaST and X-based YaST2. An older tool, GNOME RPM (or `gnorpm`) works with any RPM-based distribution. If your distribution uses another tool, you may need to consult its documentation for details; however, the basic principles should be similar to those described in this chapter.

TIP: *Although Synaptic was originally created for Debian, it and the APT utilities upon which it relies have been ported to use RPM. Therefore, Synaptic can be used with any RPM-based distribution. Check <http://apt4rpm.sourceforge.net> for pointers to ports of APT and Synaptic for RPM-based distributions. (The APT binary package works across distributions of a given CPU type, but you'll need to adjust your `/etc/apt/sources.list` file for your particular distribution.)*

One caveat concerning package browsers is that some of them (such as the Fedora/Red Hat Package Management tool) ignore packages that aren't part of the OS's installation. That is, if you install a package from a third party, it might not show up in the package browser. Most package browsers, including GNOME RPM, `dselect`, and Synaptic, will display "alien" packages that are installed on the system.

Using Red Hat Package Management

Because of the popularity of Red Hat and its Fedora variant, Red Hat's Package Management tool is very common. You can launch this program from the standard Fedora/Red Hat desktop by selecting System Settings > Add/Remove Applications from the main menu. Alternatively, typing **system-config-packages** (or **redhat-config-packages** on some versions of Red Hat) runs the program. The result is a display similar to Figure 6-1. If this window looks familiar, it may be because Fedora and Red Hat use it during system installation. It works the same way after installation as during installation.

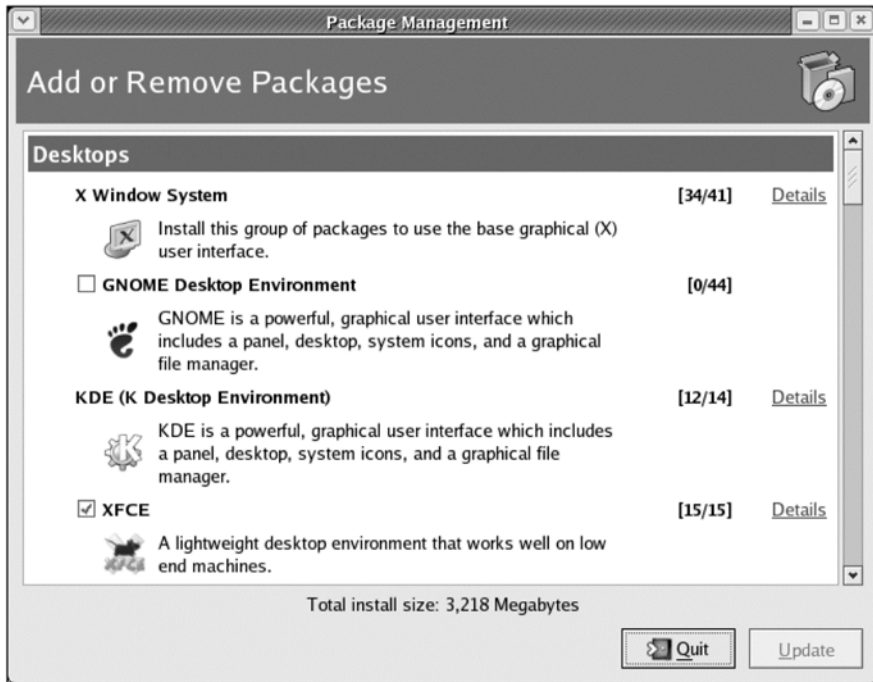


Figure 6-1

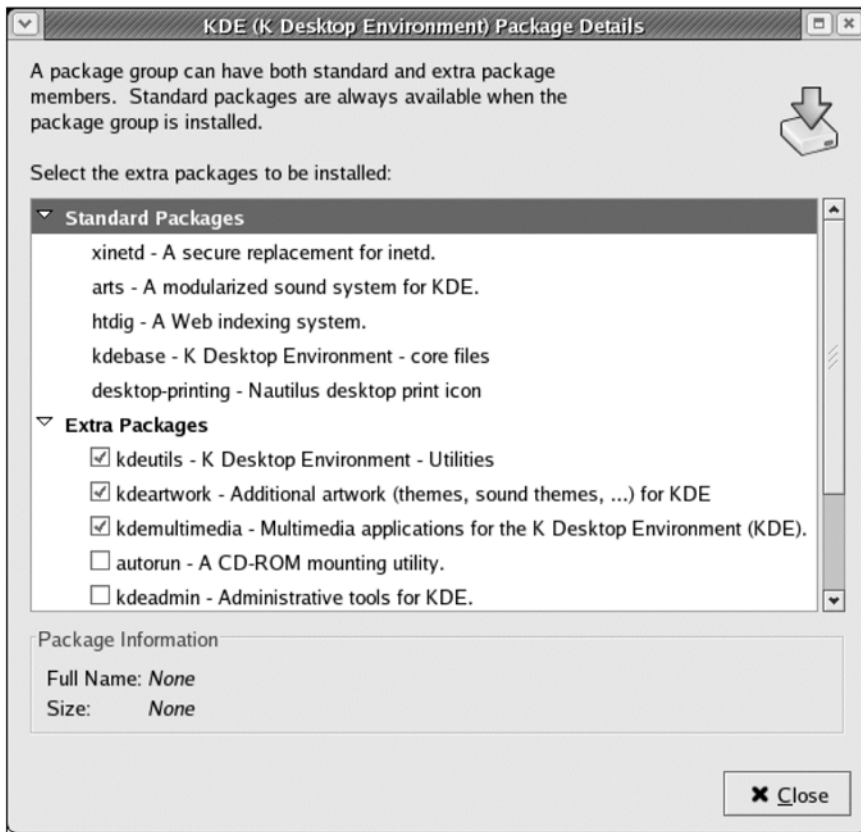
The Red Hat Package Management program provides point-and-click package management in the Fedora and Red Hat distributions.

To use Red Hat Package Management to clean out gunky packages, follow these steps:

1. Launch the program as just described.
2. Scroll through the list of program categories. (In Figure 6-1, these are X Window System, GNOME Desktop Environment, and so on.) Use the scrollbar to the right to browse through the entire list. If a category is selected and you know you don't want anything in the category, uncheck the box to the left of the name.

WARNING: *Be very cautious about deselecting an entire category. Sometimes a category, or at least a few programs in it, are important for reasons that aren't immediately obvious. Some categories can't be completely deselected because some programs within those categories are important for basic system operation.*

3. Scroll through the list of program categories a second time. This time, for each category that's selected, click the Details button. The result is a package details dialog box, such as the one shown in Figure 6-2. This dialog box

**Figure 6-2**

The package details dialog box enables you to manage individual packages.

shows the individual packages that are present in the category split into two groups: standard packages that are necessary for the group to function, and extra packages that are optional and may be added or removed at will.

4. Within the extra packages area, check or uncheck the packages you want to add or remove. When you're done, click the Close button.

TIP: Unfortunately, the descriptions presented in the Package Management tool are a bit terse; they aren't as complete as the descriptions provided by `rpm`. If you're uncertain about the purpose of a package, open a command prompt window and use the `rpm` command to view the package's full description. For instance, you might type `rpm -qi kdeutils` to learn more about the `kdeutils` package shown in Figure 6-2. If you want to learn about a package that's not installed, you'll need to locate it on your installation media and type `rpm -qpi package-filename`, where `package-filename` is the package filename.

5. When you've reviewed all your package areas, click the Update button in the main Package Management window. The system summarizes your changes and asks for confirmation. Click Continue and the program will proceed with the updates. If you elected to install new software, the program may ask you to insert your original installation media.

Of course, much of the trick to using Package Management to clean out unused packages is determining what packages are actually unused. That topic is covered shortly, in "Identify Unused Packages."

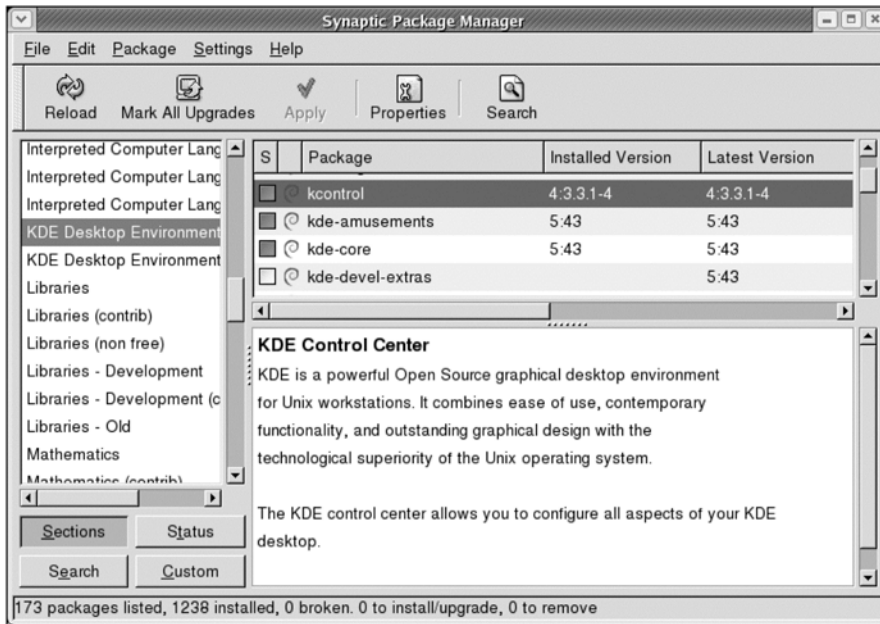
Using Synaptic

Debian's X-based Synaptic is very similar to Debian's text-mode `dselect`. These tools are also available for RPM-based distributions, but you must first install APT for RPM, as mentioned earlier, in "Options for Package Browsers." Personally, I like APT and Synaptic, even on RPM-based systems; once they're properly configured, installing and removing software is a snap, and you can easily peruse full package descriptions (not just the truncated descriptions of Fedora/Red Hat Package Management) for everything that's installed on your system. The catch is that installing APT on non-Debian systems is an extra task. If you've installed APT on an RPM-based system, typing **`apt-get install synaptic`** should install Synaptic. (On some distributions, though, you may need to track down Synaptic separately. This is true for SuSE, for instance. Try searching for it on <http://www.rpmfind.net> if you can't install it via APT.)

Once APT and Synaptic are installed, you can launch the program by typing **`synaptic`** in a command prompt window as root. (Some distributions may also create an entry in your desktop environment's menus.) The result resembles Figure 6-3, which shows Synaptic running on a Debian system.

NOTE: Early versions of Synaptic, some of which are still used with the "stable" version of Debian 3.0r4, look quite different from the one shown in Figure 6-3. The basic functionality is the same, but details of how to access certain features differ. Figure 6-3 shows Synaptic 0.55.

To use Synaptic to filter gunk out of your system, peruse the package list in the right pane of the window. The pane below the package list provides the package's text-mode description. If you want more information, such as a list of dependencies and installed files, you can click the Properties button after highlighting the package. The result is a dialog box with assorted information on additional tabs. In the main window, installed packages are marked in the S column in the package list, such as `kcontrol`, `kde-amusements`, and `kde-core` in Figure 6-3. To mark a package for removal, right-click it and select Mark for

**Figure 6-3**

Synaptic provides detailed information about both installed and uninstalled packages.

Removal from the resulting dialog box. You can mark packages for installation in the same way, except that you select Mark for Installation rather than Mark for Removal.

The complete list of packages displayed by Synaptic is quite huge. You can use a few features to trim it down to something manageable:

- ✓ Select a predetermined set of packages to be displayed by selecting that set in the sections list to the left of the package list. For instance, Figure 6-3 shows the KDE Desktop Environment section, restricting packages to those associated with KDE.
- ✓ Use the search feature. Click the Search button in the button bar or select the Edit > Search menu item. You can then type a search string in the resulting dialog box to restrict the listing to packages matching that pattern.
- ✓ You can restrict the listing to installed packages by clicking the Status button near the bottom-left corner of the window and then clicking Installed in the pane to the left of the package list. (This is the same pane that displays package sections in Figure 6-3; the list changes depending on the button below the list that you've selected.)

For a gunk-cleaning session, restricting the listing to the installed packages probably makes the most sense. The other options are most likely to be useful when you're looking for a particular package or type of package.

When you're done selecting packages for removal (or for upgrading or installation), select the **Edit > Apply Marked Changes** menu item or click the **Apply** icon in the button bar. Synaptic will call the underlying APT utilities to do the work. Depending on your APT configuration, you may be asked to insert your original installation media if you chose to install any new packages. You may also need to interact with the utility as it asks questions about the packages you're installing or removing.

Of course, as with Fedora/Red Hat Package Management, actually knowing which packages to remove can be trickier than mastering the mechanics of removing packages. This topic is up next.

Identify Unused Packages

Perhaps the most difficult part of degunking your installed packages is determining which ones are really unused. Particularly if you're unfamiliar with Linux, the package names can be quite uninformative—do you really need `pam`? (Probably.) What does the `gdb` package do? (It's a debugger—a software development tool.) Unfortunately, the number of packages is such that I can't tell you what each one does and whether or not it's necessary. In fact, the packages vary from one distribution to another! Instead, I'll provide a few tips to help you figure it out for yourself. These include some pointers on interpreting package descriptions, finding additional information on the package, testing the program, and dealing with dependencies.

Interpreting Package Descriptions

The first thing to check, and the easiest if you're using a package browser, is the package's description. This description should provide some clue to the importance of the package. This is particularly true if you're using Debian or one of its derivatives. The Debian package system provides a priority field, as described earlier; if this field identifies the package as required, you shouldn't even think about removing it. Even lower-priority packages may be vital for your system, though, so you'll need to interpret the description.

Package descriptions usually provide information on what the package does—they may describe the package as holding an office suite, a Web server, a set of fonts, or what have you. Here are some words and phrases to watch for in these descriptions:

- ✓ **Library**—A library is a set of software routines used by other programs. Library packages frequently contain the string `lib` in their names. Even if a library doesn't sound like something you need, it may be required by some programs you want, so you should be cautious about removing libraries. On the other hand, both the RPM and Debian package management systems will refuse to uninstall a library if it's used by other programs unless you provide special override options to the utilities. Thus, you're unlikely to do harm by attempting to remove a required library.
- ✓ **Headers or development**—A *header* is a file that's used to refer to libraries in source code. If you don't compile your own software, you probably don't need header packages. These packages are sometimes described as *development* packages.
- ✓ **Server**—Server programs are potentially hazardous, so for the most part, you shouldn't install them unless you intend to run the server in question. One notable exception is the *X server* (XFree86 or X.org-X11), which provides the Linux GUI environment. Some distributions, such as Fedora/Red Hat and Mandrake, also rely on the *xf86* font server.
- ✓ **Window manager**—This is a program that manages windows by providing a title bar, window resizing tools, and so on. Window managers are described in Chapter 4. In theory, you can delete all the window managers except the one you're using. You might want to keep *twm* and *fvwm* around, though; these are old but basic window managers that are often used as fallback tools. Note that window managers associated with desktop environments are often installed in separate packages, so heed any dependencies between window managers and other desktop environment packages.

Finding Additional Information

If you're puzzled by a package description, you'll have to go looking for additional information. Fortunately, a few procedures can help out a lot:

- ✓ **Check the man pages.** You can look for documentation on your local computer. Try typing `man`, followed by the package or program name at a command prompt, to read the man page for the package or program. Man pages are basic documentation, usually written in a terse style. They may provide the clues you need to determine whether a package is really necessary.
- ✓ **Consult additional local documentation.** You can look for documentation files associated with the package. Some package browsers, such as Synaptic, can provide a list of installed files. Look for files in a documentation subdirectory and start reading them. If you're using an

RPM-based system and your package browser doesn't provide this information, type `rpm -ql packagename` at a command prompt, where *packagename* is the name of the package, to obtain a list of files.

- ✓ **Perform a Web search.** Go to your favorite Internet search engine and type in the name of the package. Chances are you'll find a hit that will help you decide whether it's gunk or something worth keeping.
- ✓ **Go to the package's home page.** Sometimes package descriptions, particularly on RPM-based systems, include a URL for the software. This site is likely to contain a helpful description of what the package does.

Testing the Package

One radical approach to determining whether or not a package is gunk is to test it. For a software package, this means running the program, so you should use your package tools to obtain a list of files in the package, paying attention to those in `bin` and `sbin` directories (such as `/usr/sbin` or `/usr/X11R6/bin`). Files in these directories are executable program files, so you can type their names at a command prompt to see what they do.

WARNING: *Some programs do dangerous things. Thus, I recommend performing such a test only as a last resort unless you already have some idea of what the program does and you want to learn the details. You should also run unknown programs first as a regular user rather than as root. Many programs also accept `--help` or `-h` as a command to display basic use information, so try passing these options the first time you run an unknown program, as in `/usr/bin/someprog --help`.*

Non-program packages can't be tested in quite this way. Such packages include libraries, fonts, and documentation, among other things. You'll need to test such packages in some other way, such as reading the documentation files in a text editor or Web browser. (Some documentation files require special viewers, though, such as the `man` utility for reading man pages.) Libraries aren't easily tested unless you know what programs use them—and in that case, you should know whether the library is necessary or not.

GunkBuster's Notebook: Dealing with Dependencies

Packages can be required for one of two reasons:

- ✓ The package is actually necessary for basic system functionality or for your specific purposes.
- ✓ The package is marked as a dependency of another package. For instance, a word processor might depend on a font collection or a compression tool, so if you install the word processor, you must also install these other packages.

The first of these reasons is straightforward to understand, even if the necessity of certain system packages isn't always obvious to the uninitiated. The second reason, though, can create a confusing mess of dependencies. You might easily decide you can do without the compression tool that's required by a word processor, for instance. When you try to remove the compression package, the package management system should respond with an error message concerning failed dependencies, as in this example:

```
# rpm -e unzip
error: Failed dependencies:
        unzip is needed by (installed) xmms-1.2.10-9.i386
        unzip is needed by (installed) openoffice.org-1.1.2-
11.5.fc3.i386
```

This attempt to remove the `unzip` package (which holds the `unzip` program for extracting data from zip files) has failed because this package is required by two others: `xmms` and `openoffice.org`.

Some package browsers will give you the option to remove the packages that depend upon the one you choose to remove. If you run into such an option, be sure to study the list of additional packages that will be removed. If in doubt, abort the package removal until you can study the issue in more detail.

Keep Software Up-to-Date

Although removing unnecessary software is a fairly obvious type of package degunking, another important degunking task is keeping your software up-to-date. Before proceeding with actual software updates, you should know why performing this task is important—after all, if your system is working to your satisfaction, why bother? Linux supports various tools for performing software updates, from update options in the `rpm` and `dpkg` programs to automated GUI tools. I describe two of these tools: Fedora and Red Hat's Update Agent and Debian's Synaptic. (Synaptic does double duty as a package installer/uninstaller and as an update tool.)

NOTE: Many distributions provide distribution-specific software update tools. I can't cover them all here, so I've described these two. As noted earlier in the chapter, Synaptic has been ported to work with most RPM-based systems, so if you like, you can install it on Fedora, Mandrake, Red Hat, SuSE, or other RPM-based systems and use it. This is my own preference with such distributions, although it does require you to install and configure APT for RPM first. Check <http://apt4rpm.sourceforge.net> for more information.

Why Keep Software Up-to-Date?

Chances are your Linux system is working reasonably well. Thus, you might adopt an “if it ain’t broke, don’t fix it” attitude to system updates. In truth, there is merit to such an attitude—software updates can sometimes cause more problems than they solve. After all, modern OSs, Linux included, are something like houses of cards, with each card being a software package. By upgrading one package (particularly one near the base of the house), you run the risk of having everything else tumble down around you.

Fortunately, such disasters are rare, and when problems do occur, you can usually fix them with a bit of troubleshooting effort. There’s also something to be said for upgrading your system, even if it seems to be working correctly. Such upgrades can have several important effects:

- ✓ **Security updates**—The most important reason to keep your software up-to-date is to receive security updates. These are fixes to bugs that can give outsiders access to your computer, give local users root privileges they shouldn’t have, or otherwise compromise the integrity of your computer.
- ✓ **Non-security bug fixes**—Other bugs aren’t really security issues, although some can be serious. Most bugs are likely to affect just one program, but bugs in popular libraries can affect many, and kernel bugs could even crash the computer. Some bugs are sporadic in nature, so you might not have encountered them—yet. If you don’t upgrade to a fixed package, you might run into the problem tomorrow.
- ✓ **Software improvements**—Linux software is constantly being improved. While your system might be working to your satisfaction right now, an improvement might be available that would make it work even better. What’s more, chances are you have some cause for complaint about your system. Perhaps a program seems a bit sluggish or another program lacks a feature you really want. Occasionally an update will make the program work more to your liking.
- ✓ **Dependencies**—Old software can keep you from installing new software because of conflicting dependencies. Thus, upgrading your existing software can simplify the process of adding software to your system.
- ✓ **New software**—Although new software won’t be installed automatically unless it’s a dependency of an updated package, new programs do become available from time to time. If you keep your package database up-to-date, such programs might appear as options (particularly if you use Synaptic or another package tool that keeps a local database of available software). You might not notice such options immediately, but if you peruse the options or otherwise monitor Linux software developments, you should find them and be able to install them as you see fit.

These reasons are ordered by decreasing importance. The most important reason to keep your system up-to-date is to improve its security. This is particularly important if your system is directly connected to the Internet, and especially if you run a popular server (a mail server, a Web server, and so on). Security bugs in such servers, once discovered, are rapidly exploited by crackers, so you should keep on top of things to avoid serious problems.

Using Update Agent for Software Updates

If you use Fedora or Red Hat, your system has an automatic software update tool installed: Update Agent (aka `up2date`, the name of the program executable). If you run Fedora or Red Hat using either GNOME or KDE as your desktop environment, Update Agent runs in the background and displays an icon near the lower-right corner of the screen, as shown in Figure 6-4. (This location can be changed depending on your desktop settings.) A red circle with a blinking exclamation mark, as shown in Figure 6-4, indicates that updates are available. If you move your mouse over the circle, the system displays the number of updates available—Figure 6-4 shows that 65 updates are available.



Figure 6-4

Update Agent makes it obvious when system updates are available.

To actually perform an update, you can either click on the red Update Agent icon in the Panel or type `up2date` in a command prompt window. The system will ask you for the root password and then display the main Update Agent window, which works like the “wizards” that are used to install software on Windows—the window provides settings and Forward and Back buttons to enable you to adjust settings and move through the process of, in this case, installing updates. Once you move past the first introductory page, Update Agent asks you which *channels* you want to use, as shown in Figure 6-5. Typically, the defaults are good choices, so you should leave this option unchanged.

Once you select the channels you want to use and click Forward, Update Agent downloads information on the available updates. This process can take anywhere from a second or so to over an hour, depending on the number of updates, your network connection speed, and the speed of the archive site. You may be told about packages that will be skipped because of prior selections

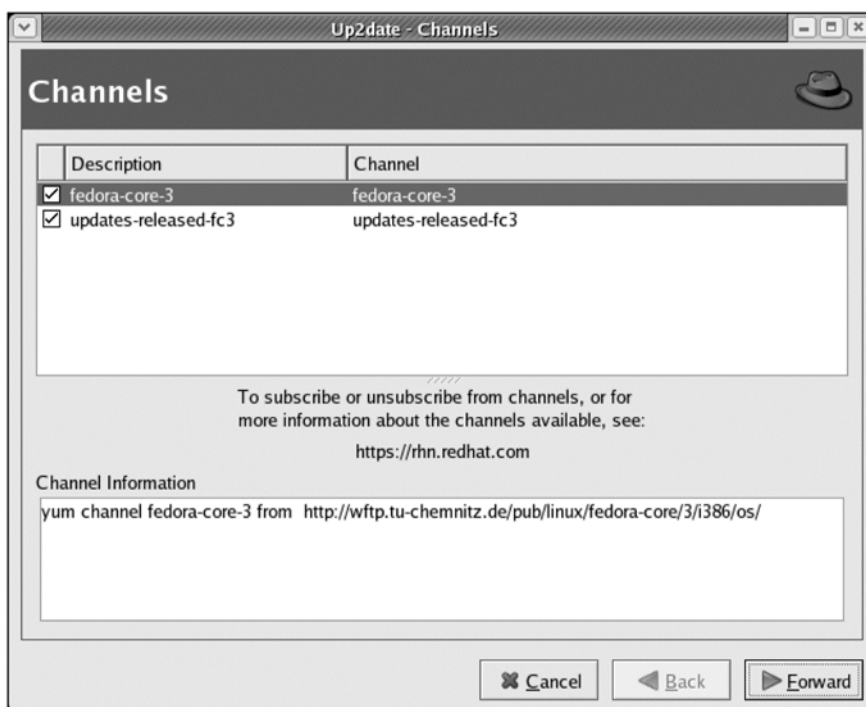
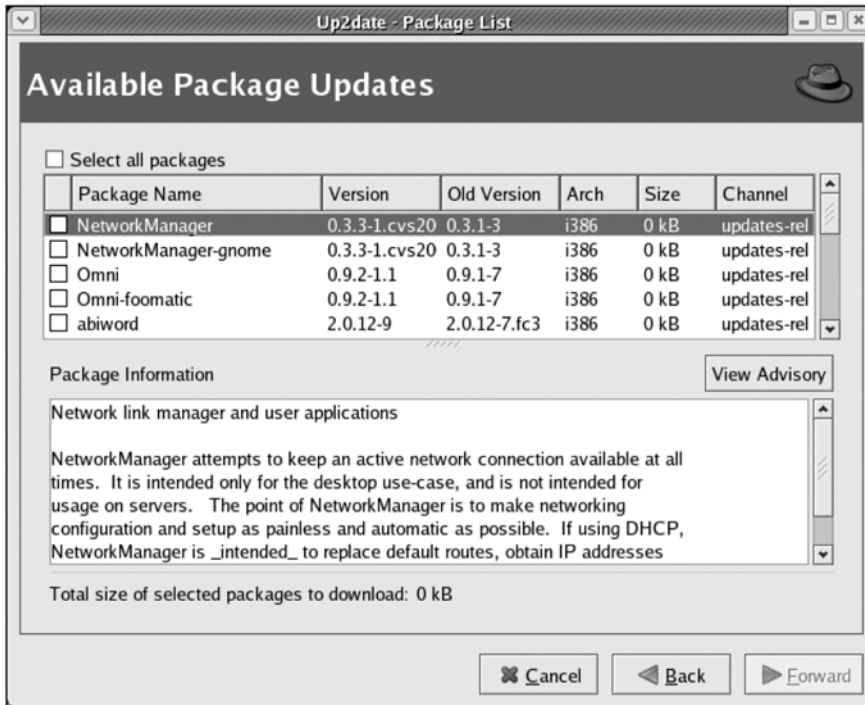


Figure 6-5

Update Agent can pull software from one or more repositories, which it calls *channels*.

you've made. (Some packages—notably kernels—can be trickier to upgrade than others, so you might opt to bypass upgrading them until you're sure you want to do so.) Update Agent then displays a list of available updates, as shown in Figure 6-6. You can click an update to view a description of it. Select the packages you want to update by checking the boxes to the left of their names, or check the Select All Packages box to select them all. When you click Forward again, the system does some housekeeping chores, downloads the packages, and installs the packages.

If you opted to upgrade all of the packages for which upgrades are available, the red Update Agent icon in your Panel (Figure 6-4) will change to green. If not, it will remain red, which can be an annoyance but isn't a major problem, except that it won't then alert you to the presence of new updates unless you move your mouse over it, whereupon you'll see that the number of available updates has increased.

**Figure 6-6**

You can review updates and add or exempt packages from the upgrade operation before proceeding.

Using Synaptic for Software Updates

Synaptic's basic features are described earlier in this chapter, in "Using Synaptic." Using the tool for software updates is similar to using it for cleaning out gunky packages; you just select somewhat different options. In particular, to begin the system update process, you should click the Reload button in the button bar. This action tells Synaptic to contact the package archive sites you've defined in `/etc/apt/sources.list` to download the latest package descriptions. This process will take several seconds, during which time a progress dialog box appears on the screen.

Once you've reloaded your package descriptions, click Mark All Upgrades in the button bar (see Figure 6-3). The program asks you if you want to upgrade the system in a "smart" way. The default method skips upgrades that are likely to cause dependency problems, but a smart upgrade will upgrade such packages if it can resolve any dependency problems they create. The default method is good for a quick upgrade if you don't want to risk problems, but a smart upgrade may be necessary if an upgraded package includes new dependencies (say, because the developers have begun using a new or upgraded library).

*NOTE: Synaptic performs the same tasks as `apt-get`. In particular, a default upgrade is equivalent to typing **apt-get upgrade** and a smart upgrade is equivalent to typing **apt-get dist-upgrade**. You can type these commands at a command prompt if you prefer to do this rather than use a GUI tool. Assuming Synaptic finds packages that can be upgraded, the Apply button in the button bar will become available. You can click this button to tell the program to retrieve the updates and install them. Before doing so, the system displays a confirmation dialog box in which all the upgrades are listed, as shown in Figure 6-7. The initial dialog box is likely to only contain summary lines, such as To be upgraded and To be installed. To see a list of packages within each group, click the triangle to the left of these summary lines. These lists only provide the package names (and package versions, if you click the Show Details button). If you want to find more details about the packages that are to be upgraded, you should cancel the operation and search for those packages to view their configurations.*

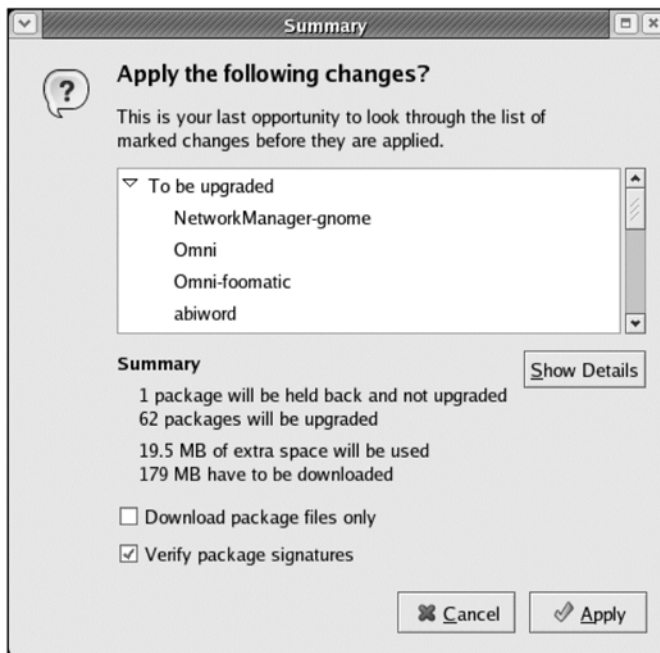


Figure 6-7

Like Update Agent, Synaptic summarizes the updates before it performs the operation.

Handle Dependency Problems

Throughout this chapter, I've referred to dependencies. In many ways, dependencies are the thorn in the side of Linux users when it comes to package management, because dependencies can result in a need to upgrade a dozen packages just to upgrade one, block removal of a package that you consider gunky, and sometimes cause even worse problems, as described shortly. Fortunately, advanced package management tools like APT, Synaptic, Red Hat Package Management, Update Agent, and YaST/YaST2 can greatly simplify managing dependency problems; however, sometimes you may need to break a few rules. Doing so can work wonders if you exercise caution and know what you're doing, but it can wreak havoc if you make a mistake.

Tracking Down Dependencies

If you're using RPM or Debian packages, you usually don't need to go looking for dependencies; they'll make themselves known when you try to install, remove, or upgrade a package. If you use the raw `rpm` or `dpkg` tools and you try to do something that breaks a dependency, you'll get an error message. It's then your job to fix the problem. Depending on what you're trying to do, this fix can take any of several forms:

- ✓ **Delete extra packages.** If you're trying to remove a package but find that it's depended upon by other packages, you can remove those packages as well. Of course, you should be sure those other packages aren't ones you really need!
- ✓ **Upgrade or install extra packages.** If an attempted upgrade or install fails because of a missing or out-of-date dependency, you'll need to track down an appropriate package and upgrade or install it in addition to the one you want to upgrade or install. In practice, you might find you need to upgrade or install dependencies of the dependency and so on, resulting in a need to deal with many packages—perhaps dozens of them. This is the sort of situation that begs for the use of APT, Update Agent, or a similar tool to help keep track of all the interlocking dependencies!
- ✓ **Look for another package.** Sometimes a package you want to install requires a package that's older than the one you've got installed. For instance, the package might say it needs `superlib-1.2` but your system has `superlib-1.9`, and some of your existing programs require this version of the library. Usually, the best fix for this problem is to look for a newer version of the package you want to install, because this problem is a sign that you're trying to install an old version of the program. If the program hasn't been updated in a long time, perhaps it's time to switch to another program entirely.

NOTE: If you're an advanced user, you may be able to recompile an old package to use newer libraries. With RPMs, this can be done by downloading a source RPM, which is an RPM that contains source code rather than compiled code. You can then compile the source RPM by typing `rpmbuild --rebuild package.src.rpm`, where `package.src.rpm` is the source RPM file. The result is a fresh binary RPM, located somewhere in the `/usr/src` directory tree, that you can install. (The exact location depends on your distribution and CPU type; try using the `find` command described in Chapter 3 to locate it.) This process requires that you have various development tools installed, and it may fail for various reasons. You might give it a try if you're really desperate to get an older or obscure package working on your system.

If you stick to software that's supported by your distribution, you're unlikely to run into really bad dependency problems. At the worst, you should be able to use package management tools like Synaptic to keep everything in sync, and you may need to swap installation CD-ROMs in and out of your CD-ROM drive as it asks for them.

Serious dependency problems are more likely to arise if you start using software that's not been vetted by your distribution maintainers. Such packages may be built for a different distribution than the one you're using—for instance, the packager might have used Red Hat when you're running Mandrake. Most of the time, such packages install and run fine, assuming they're in the correct format (such as RPM). Sometimes, though, dependency conflicts arise, complicating installation of the software or subsequent upgrades. If this happens, you'll need to try the procedures just described to resolve the problem.

TIP: If you're using an RPM-based distribution, you can check <http://www.rpmfind.net> for many packages. You may be able to use that site to find a newer version of a problem package, missing support packages, or other packages that can help you work around the problem. Making heavy use of this site, though, can lead you deeper into eventual dependency problems, as the packages found there may not be registered with your distribution and so might not be easily upgraded with the advanced package management tools.

Bypassing Dependencies: At Your Own Risk

By far the best way to deal with a dependency problem is to resolve it within the package management system by installing, upgrading, or removing packages, as just described. Sometimes, though, you can bypass dependencies by telling the package system to ignore the problem. This approach can work if the specified dependency is overly stringent. For instance, a word processor might list a dependency on a set of fonts simply because the program author thought people using the word processor should have a good selection of fonts, not because the word processor actually *requires* the fonts. When this is the case,

bypassing the dependency can make sense. Similarly, you might choose to ignore dependencies if you know they're met in some other way—for instance, if you've installed a library by compiling it from source code rather than installing it from its own package. (Generally speaking, you're better off using packages rather than compiling software yourself with most Linux distributions, but you might have a reason to compile a program locally.)

To override a dependency, you can use advanced command-line options to the `rpm` or `dpkg` programs. For `rpm`, you can use the `--nodeps` option with the `-i` (install), `-U` (upgrade), or `-e` (erase) option:

```
# rpm -Uvh --nodeps overdependent-1.23-4.i386.rpm
```

This command installs the overdependent package even if it depends on packages that aren't installed on the system. For Debian-based systems, the `--ignore-depends=package` option works in a similar way, but you must specify the name of the package with the option:

```
# dpkg -i --ignore-depends=overdependent overdependent-1.23_4.dpkg
```

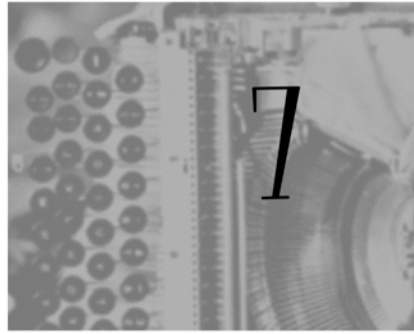
Ignoring dependencies in this way can help you out of certain tight spots; however, if used inappropriately, this option can cause the program to fail. Perhaps a program won't run at all or will fail to perform certain actions it should handle. Just as bad, advanced package managers such as Synaptic are likely to throw a fit when they find their systems are “contaminated” in this way. The package managers may refuse to do anything until the offending program is removed or its dependencies are met. Even if you're willing to live with such problems, future upgrades of this or other programs may fail because of these unmet dependencies. Thus, whenever possible, you should play within the rules of the package management system, rather than try to work around those rules.

Summing Up: Removing Package Gunk

Package gunk comes in two main forms: unused packages and old packages. Unused packages consume disk space, pose a security risk, and can cause headaches down the line because of their dependencies. Old packages can cause problems because of their bugs (particularly their security bugs) and dependency problems. You can deal with both types of gunk with the help of Linux's package management systems and particularly advanced package browsers like Synaptic and Red Hat Package Management. These tools help you add, remove, and upgrade software, keeping your system slim and up-to-date.

Summing Up: Keeping Gunk Out of Your Applications

Linux applications, like Linux as a whole, can accumulate gunk. As a general rule, the larger and more complex an application, the more likely it is to become gunked up. Three of the largest and most popular types of programs in desktop Linux use are office suites, Web browsers, and e-mail clients. This chapter describes the type of gunk that can build up in these applications' configurations and how to deal with it, focusing on the largest or most popular example of each type of program. As a general rule, a check of a program's settings can go a long way toward fixing any problems that exist. Another good rule of thumb in degunking applications is to attend to potential security threats. If you're unsatisfied with a program's performance, you should also look into alternatives; in many cases, a smaller and faster program can work as well for you as a big one.



Improving Software Performance

Degunking Checklist:

- ✓ Understand the causes of poor system performance.
- ✓ Take steps to improve the speed of desktop applications.
- ✓ Take steps to improve the reliability of programs.
- ✓ Tweak your printer configuration.

If one thing has been consistent over the past few decades in the computer world, it's been the steady improvement in the speed of computers. Over the past two decades, computer CPU clock speeds have improved by a factor of about 500 (actual computational speed has gone up by even more), while available RAM has increased by a factor of about 1,000. Despite these improvements, today's computers can seem quite sluggish at times. As detailed shortly, this sluggishness has several causes. This chapter is devoted, in part, to improving this situation by helping you fine-tune your computer's software. Another topic in this chapter is improving software reliability; a crashing program is no fun, and although there's no silver bullet for avoiding crashes, you can take some steps to improve software reliability. Finally, this chapter looks at printing performance, because getting output is a common bottleneck in the Linux software chain.

What Causes Sluggish Performance?

In the best of all possible worlds, computers would finish their work instantly—click a button to recompute something and it's done; press the key to copy a file and it's done. Sadly, this isn't the case. Computers take time to perform their work. Sometimes the time periods involved are very small by human standards, but other times they aren't. The causes of less-than-optimal performance are varied, but they can be classified in four main ways: software problems (that is, problems with the software code itself), configuration problems (which affect software because the software must do more work than is really necessary), interference from other programs, and hardware problems. Sometimes the problems in these categories interact with one another, exaggerating modest problems in each category to the point that they become real issues. Understanding these problems will help you identify and avoid them when you tackle a sluggish system.

Software Problems

The first broad class of problems that causes sluggish performance is software. Compared to the software of yesteryear, today's software is very inefficient, for several reasons:

- ✓ **Language choice**—Today's software is almost always written in C, C++, and other *high-level languages*. These computer languages are easy for humans to understand, but they must be *compiled*—that is, translated into the binary machine code that computers understand. This compilation process usually produces code that's larger and less efficient than the handcrafted machine code that was more common 20 years ago. There's not much you can do

about this issue, although recompiling your software with optimum settings for your CPU might produce a modest speed boost. (This is one of the claimed benefits of the Gentoo Linux distribution.)

- ✓ **Sloppy coding**—Even within a given computer language, programs can vary in their efficiency. This is partly the result of the quality of the program source code. As with program language choice, you really have no way to control this matter short of modifying the software yourself. You may be able to locate more efficient software, though.
- ✓ **User interfaces**—Computer users have been demanding more and more sophisticated user interfaces. The original Macintosh popularized the GUI, and in the years since, GUIs have become more and more complex. Each step up in complexity comes at a cost in performance, though, because the code to support it becomes larger and consumes more CPU time. Your main defense against speed degradation as a result of such improvements is to use programs that sport simpler user interfaces. Sometimes you can recompile a program using options to disable the more CPU-hungry user interface features, but this is a very advanced topic.
- ✓ **Complex features**—Even aside from the user interface, modern programs support more features than ever. This support comes at a cost, even if you're not using the features. The program itself becomes larger and its code becomes more complex. As with user interface complexity, the best way of protecting against these problems is to choose simple programs whenever possible. For instance, use the moderately sized Abi Word rather than the behemoth OpenOffice.org for word processing.
- ✓ **Executable size**—A factor that's the result of many of the former factors but that has an effect of its own is code size. Larger programs consume more memory, which can cause more memory swapping on systems with insufficient RAM. Even on systems with lots of memory, large programs take longer to load from disk than small programs do.

The combination of these factors is known as *code bloat*—the tendency of programs to become larger and less efficient as they're upgraded. For the vast majority of programs, version 1.0 is smaller and more efficient than version 1.1, which is smaller and more efficient than version 1.2, and so on.

Fortunately, the increases in CPU speed, RAM size, and other hardware speed improvements serve to counter code bloat; however, if you don't upgrade your hardware, code bloat can become a serious problem. One defense, of course, is to stick with older or simpler programs. Linux can help with this. Linux programs usually follow in the Unix tradition of piecing together several smaller programs to do a larger task, which means you can tweak your software to omit those

parts you don't need. One potentially dramatic example is with desktop environments, as described in Chapter 4, "Degunking Your Desktop Environment." Rather than use the full GNU Network Object Model Environment (GNOME) or K Desktop Environment (KDE), you can use a slimmer desktop environment such as XFce or even a bare window manager. You can then add just those components you need from independent projects or even from the GNOME or KDE projects. This approach won't always work, though; for instance, you can't really slim down OpenOffice.org very much. You can switch to an alternative office suite, though.

If you're faced with a bloated program that's causing you problems, you should look for alternatives. One good place to begin such a search is Linux Online's applications area (<http://www.linux.org/apps/>). You can search for applications in each of many categories on this site. With luck, you'll find one that's not as bloated as the one that's giving you trouble. Once you've located likely replacement programs, you should try using your distribution's package management tools (such as those described in Chapter 6) to install the software. If the package isn't available as part of your distribution, you may need to download it from the author's site.

Configuration Issues

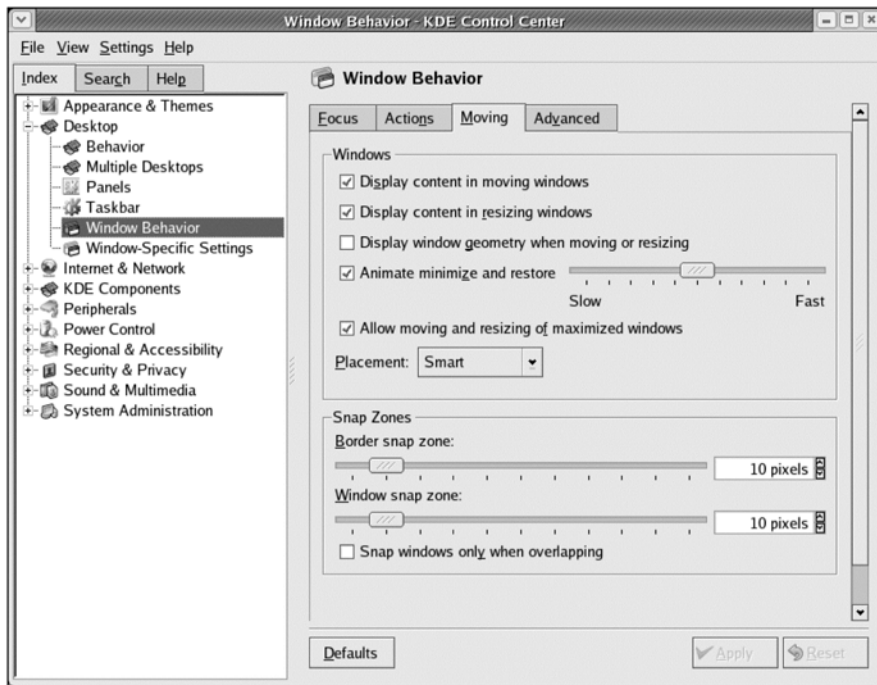
A second broad class of performance-degraders is configuration issues. These include both system configurations and account configurations. Several likely sources of problems exist:

- ✓ **Running too many programs**—This issue overlaps substantially with the next broad category, described in the section "Interference Issues." If you run too many programs simultaneously, they'll consume memory and CPU time, taking it away from your primary program. As a user, you can simply close down programs you're not actively using if your system becomes sluggish. As an administrator, you can track down and uninstall or kill unnecessary programs. Chapters 6 and 8 can help with these tasks.
- ✓ **Poor program options**—Some programs can be launched using options that influence their performance. This issue is very program-specific, though. If a particular program is behaving poorly but others are okay, you should consult the documentation for the sluggish program.
- ✓ **X server options**—Your X server plays an important role in overall system performance, at least for desktop systems. Poor X server options can severely degrade the speed of your system. Two options in particular are vital. The first is picking the correct X driver. Most video cards can be driven via generic drivers that work on a wide array of cards but produce

poor performance. A better choice is a driver for your specific video card (or more precisely, its chipset). A second important X option is the color depth. This is usually 8, 16, 24, or 32 bits, corresponding to 256 colors, 65,536 colors, 16,777,216 colors, or 4.29 billion colors. Although more colors can look good, boosting the color depth also slows performance on video-intensive tasks. For most purposes, going beyond 24-bit is pointless, and 16-bit may be sufficient. These and other X configuration issues are covered in greater detail in Chapter 12, “Optimizing Your X Configuration.”

- ✓ **Font issues**—Fonts can severely degrade performance, particularly in font-intensive programs. The main problem is when you’ve installed too many fonts. Certain programs, such as OpenOffice.org, attempt to render all the fonts so you can see what they’ll look like when you choose a font. The result can be sluggish performance when selecting fonts. The usual solution is to uninstall the fonts that you don’t need. Consult Chapter 6 for information on removing font packages.
- ✓ **CPU-guzzling eye candy**—KDE and GNOME both provide a plethora of options that are designed to make them more friendly or visually interesting. Such options are generally referred to as *eye candy*, and they can look nice. Unfortunately, they also consume CPU time. Examples include cursors that bounce when you launch programs, animated icons, and so on. You can go into the KDE or GNOME control panels on a search-and-destroy mission to disable such options. Some programs unaffiliated with KDE or GNOME have similar options, so peruse your programs’ configuration options to find them.
- ✓ **Multimedia features**—Increasingly, multimedia is taking center stage, even with traditionally non-multimedia applications. Such options, like eye candy, can be nice luxuries, but if your hardware isn’t up to handling them, they’ll degrade performance unnecessarily. As with other performance-degrading features, you should look for ways to disable such features.

If one word dominates the quest for efficient optimization, it’s *simplify*. Look for options that simplify things—fewer advanced options, fewer running programs, fewer fonts, and so on. If you use KDE or GNOME (or KDE or GNOME programs in another desktop environment), many of these options can be disabled from the KDE or GNOME control center. For instance, Figure 7-1 shows the KDE Control Center, displaying several options that can degrade performance. Specifically, the Display Content in Moving Windows and Display Content in Resizing Windows options can cause sluggish performance when you move or resize windows. Disable these options if these actions seem slow.

**Figure 7-1**

Desktop environments and other programs often provide configuration options that can influence system performance.

Interference Issues

Sometimes one program affects another one. The most common cause of such interference is conflicting demands for limited system resources—most commonly CPU time or RAM, but sometimes disk access, network access, use of a scanner, or something else.

Chapter 8, “Managing Processes,” describes how to manage running programs (also known as *processes*) so that they aren’t likely to cause problems for other processes. In brief, if a program must be running but should not interfere with other programs, you can reduce its priority in any of several ways. When the kernel dishes out CPU time, the low-priority process will take a back seat to any higher-priority processes. In this way, you can run even CPU-intensive programs in such a way that they won’t noticeably impact system performance.

Rationing other system resources, such as RAM and disk access, isn’t as easy as rationing CPU time. Fortunately, Linux’s automatic scheduling systems usually do an acceptable job at this task. The main exception comes when you simply don’t have enough of the resource in question. In this case, you must either

terminate one or more of the competing programs or upgrade your hardware. (Chapter 8 describes how to kill misbehaving programs.)

Hardware Issues

The final class of performance degraders is inadequate hardware. Of course, this is a somewhat relative matter. Hardware that's inadequate for running, say, KDE, OpenOffice.org, the GIMP, and Mozilla may be just fine for running, say, IceWM, Abi Word, XV, and Lynx—in other words, slimmer programs that might do just as well as their “big brothers,” depending on your specific needs.

Broadly speaking, you can use Linux with slimmer hardware than is needed for the latest version of Windows, particularly if you're willing to forgo the flashiest toys. Hardware requirements do tend to keep going up; however, in mid-2005 a comfortable Linux system might include the following:

- ✓ **CPU**—Linux requires an 80386 CPU as a bare minimum, but for desktop use with modern applications, count on at least a 1GHz Pentium 4 or its equivalent. Something in the 2–3GHz range will work even better. For cutting-edge use, or if you run very CPU-intensive applications, consider a 64-bit CPU such as an AMD Athlon 64. Systems with two or more CPUs can provide a speed boost if you regularly run several CPU-intensive programs simultaneously. Almost anything you buy new today will be more than adequate for basic Linux use, although of course you might have a specific need for something more.
- ✓ **RAM**—A slim desktop Linux system should have at least 128MB of RAM, but 512MB or more is a practical necessity if you intend to run the latest big programs, manipulate large files (such as high-resolution photos or audiovisual files), or run other RAM-intensive programs.
- ✓ **Disk space**—A basic Linux desktop system can install comfortably in just 1–2GB of disk space, but for plenty of space to grow, count on 6–10GB of space for Linux alone, plus however much you need for your own files. If you intend to store or manipulate big files, plan on getting at least an 80GB hard disk. Fortunately, big disks are common and inexpensive today.
- ✓ **Video cards**—Most Linux applications are happy with whatever video hardware they're given, assuming the hardware is compatible with your X server. (X server compatibility is described in more detail in Chapter 12.) Today, expensive video cards are usually expensive because of elaborate 3D engines, which are most important for games, and particularly for Windows games. If you're a gamer, you should look into Linux driver availability and features—note that the existence of a basic X driver for a card doesn't necessarily mean that its 3D features are well supported. If you're not a big game player, just about any modern video card will do nicely.

- ✓ **Monitors**—Linux isn't fussy about monitors; however, many Linux programs work best if your monitor is capable of displaying at least 800x600 pixels, and some require 1024x768 or bigger. Almost all modern monitors can handle such displays when paired with modern video cards, so this is likely to become an issue only for very old monitors.
- ✓ **Mice**—Linux has drivers for just about all mice. The only tricky thing is that many Linux programs assume the mouse has at least three buttons. If your mouse has just two buttons, you can configure X to *chord* its two buttons to simulate a third button—that is, if you press both buttons simultaneously, Linux treats it like a middle-button press on a three-button mouse. This operation is a bit awkward, though, so if you've got a two-button mouse, you might want to consider replacing it with a three-button model.

Although inadequate hardware can degrade performance, just about any system sold in the last three years or so should run a modern Linux distribution just fine. (Upgrading two-button mice may be desirable, though.) Systems that are up to five years old may also be usable but are likely to require upgrades to their RAM and hard disks. CPU upgrades are available for some computers, but by the time a CPU needs upgrading, many other systems usually need upgrades as well, making it simpler (and perhaps more cost effective) to replace the computer.

If you've got a particularly old system, want to run Linux on it, and don't want to invest in significant hardware upgrades, consider running Linux without running KDE or GNOME. Instead, run XFce or a bare window manager. You might want to consider running Debian or Slackware on such a system rather than Fedora, Mandrake, Red Hat, or SuSE. Debian and Slackware tend to install less performance-degrading gunk and so are likely to require less post-installation tuning to work well on a slim system.

GunkBuster's Notebook: Recompiling Software to Improve Performance

Most performance-tuning measures relate to changing your system configuration in some way. One way to improve performance, though, involves rebuilding your system in a very fundamental way. Most Linux software is written in C, C++, or other high-level computer languages. This software must be compiled to be used, as noted earlier in this chapter. The compilation process converts the source code into binary code that can be understood by the computer. The trouble is that the resulting binary code may not be optimal for any given system. When a distribution

maintainer, such as Red Hat or Debian, compiles the code for a binary distribution, the maintainer makes certain assumptions about your system. These assumptions are designed to make the software run on the widest range of computers possible. For x86 systems, this typically means that the compiled software will run on everything from an 80386 to the latest Pentium 4 and Athlon 64 systems, but it won't run optimally on any of them (except perhaps the oldest ones). Other assumptions involve using particular sets of libraries, building in support for particular features, and so on.

The alternative is to compile software on your own computer, ideally with options to make the software run best on *your* computer. If you do this, your software will see a modest speed boost because it can use CPU-specific optimizations. With some packages, you can also opt to include support for those features you need and exclude support for features you don't need.

In most cases, implementing system-specific optimizations like this is a tedious undertaking—so tedious that you'll spend far more time trying to do it than you can hope to regain in system performance improvements. There are a few exceptions to this rule, though. First, optimizing the Linux kernel can make some sense because *everything* relies on the kernel, so optimizing it helps to optimize everything else. Chapter 11, "Finding Drivers for Your Hardware," provides a very brief introduction to this topic, but it's really a rather advanced one.

Second, recompiling CPU-intensive programs that you use a lot can make sense. This is most likely to be true if you're running CPU-intensive scientific simulations or the like, in which case you'd probably compile them yourself anyhow. Recompiling a word processor or Web browser is unlikely to provide any real benefit.

Third, the Gentoo Linux distribution (<http://www.gentoo.org>) is built around the idea of recompiling everything. Rather than provide binary packages, Gentoo provides scripts to download and compile source code locally. In theory, this gives Gentoo a modest speed boost over other distributions. In practice, the effect is tiny, particularly if you're not familiar enough with compiler optimizations to design an appropriate optimization regime. Because Gentoo compiles just about everything locally, setting it



up and maintaining it can take a lot of time, particularly on systems with weak CPUs. Overall, you're best off avoiding it unless you're already something of a Linux expert.

Improve Program Stability

So far in this chapter, I've focused on speed as a performance factor. Other issues can be considered performance issues, though. Of these, the stability of programs—that is, their resistance to crashing—is particularly important. When a program crashes, you generally lose whatever work you've done between the last time you saved your work and the crash. Restarting the software and regaining your train of thought can also take some time. Programs can crash for many reasons, and each potential cause has several possible fixes. When a fix isn't available, you may be able to work around the problem (say, by avoiding the conditions that cause a crash). Finally, in terms of long-term program stability, you should report program crashes to their authors; this way, the bugs that cause the crashes can be fixed, never to trouble you again.

Why Do Programs Crash?

The simple answer is that programs crash because of bugs. Several causes and manifestations of crashes exist, though, each with its own unique characteristics:

- ✓ **Abnormal termination**—In some sense the “purest” type of crash is an abnormal termination. That is, the program is chugging along when suddenly and for no apparent reason, it stops. If the program has open windows, they close. If you launched the program from a command prompt, you get a prompt back, often with a message to the effect that the program ended abnormally.
- ✓ **Program hangs**—Another type of crash is a program *hang*, in which the program stops responding to input. Unlike a crashed program, a hung program keeps its windows open and retains control of any console from which it was launched. The program might as well have crashed outright, though; it doesn't respond to input and won't save its data. Hung programs often consume inordinate amounts of CPU time, unlike crashed programs, so they can interfere with normal system operation. Sometimes it's hard to tell a hung program from a program that's simply performing a long CPU-intensive computation. You'll need to be familiar with a program's normal operation to know whether, say, a two-minute period of unresponsiveness is normal.
- ✓ **Zombie processes**—Computer geeks being what they are, it's inevitable that terms like *zombie* be adopted as official nomenclature. All processes

have *parent* processes—the parent launches the child process. When a child process terminates, its parent is supposed to take care of some housekeeping tasks to remove the child from Linux’s process table. When the parent fails to do this, the result is a zombie. Zombies show up in process listings, but they don’t normally consume CPU time, so they’re relatively harmless. Zombies can usually be killed by killing their parent processes, but that may not be practical. (You might want to keep the parent process running.)

- ✓ **System crashes, hangs, or reboots**—Sometimes Linux itself crashes, hangs, or reboots spontaneously. This can result from a bug in the kernel or a hardware fault (as described shortly). Fortunately, such problems are very rare. Sometimes the system will become unresponsive because a program has crashed or hung in such a way that it ties up the keyboard and mouse. If this happens and if you’re running a network login server, you may be able to recover the system by logging in remotely and terminating the offending program. (Chapter 8 describes how to kill processes.)
- ✓ **Crashes caused by hardware failure**—This class of crashes isn’t really distinct from the preceding ones; rather, hardware failure can be the cause of any type of crash. Most commonly, bad RAM can cause all sorts of weird behavior. Sometimes a bad CPU or other system component can cause problems. Hardware problems can be notoriously difficult to track down and fix. If you suspect your hardware is causing problems, you can run various diagnostic tools. Memory-check programs, such as Memtest86 (<http://www.memtest86.com>), are a good place to start. Most hard disk manufacturers have DOS-based hard disk diagnostic tools; check your disk manufacturer’s Web page for details. (These tools, although they run from DOS, usually ship with a minimal DOS boot floppy disk image, so you can run them on a Linux-based system by rebooting with the DOS boot floppy.)
- ✓ **Crashes caused by missing libraries**—As with hardware failure, missing libraries are a root cause of a crash, not a symptom of a crash. Missing libraries typically cause a program to not run at all, and if you run the program from the command line, you’ll see a message to the effect that a particular library is missing. The solution is to track down and install the library. (Try a Web search on the name of the missing library, or look for packages with similar names in your distribution’s package manager.) In theory, if you use your distribution’s package system exclusively and don’t tell it to override its dependency checks, you should never run into this type of problem. In practice, it may crop up from time to time, particularly if you play “hard and fast” with the rules, as described in Chapter 6.

Most of these crash types will result in lost work. (Zombies are a likely exception to this rule.) Unfortunately, short of digging into the source code to fix a bug, there’s little you can do to fix the fundamental cause of a bug-induced crash or

hang. Hardware problems can be fixed by replacing the offending hardware, but tracking this down can be tricky—it can be hard to differentiate between bad RAM, a bad CPU, and other hardware problems.

If you suspect a program bug is causing problems, you should begin by looking for a newer version of the program. Chapter 6 provides information on keeping your system up-to-date; these procedures will help you update your software. You might also want to check the software's home page in case it provides an update that's not yet available via your distribution. Sometimes *downgrading* a program will fix a problem. Bugs, after all, have their origin points, and they often begin with new features or fixes to other problems, so by moving back in the software's history, you may be able to fix the bug.

Working around a Crash

One way to deal with crashes in the short term is to work around them. Sometimes the way to do so is obvious. As in the joke about the patient who complains to a doctor, "It hurts when I move my arm" and the doctor replies, "So don't move your arm," you may be able to avoid using a feature of a program that causes a crash. This approach is workable when the feature in question is something minor that you're unlikely to use, or if the program provides another way to achieve the goal you desire. Other times, though, the feature is one that's really vital—for instance, a file save option in a word processor.

If you're faced with a bug that's causing problems, you can try several ways to work around it:

- ✓ **Figure out what triggers the crash.** The first step you should take is to try to determine what causes a crash. Is it loading a file? Is it resizing the program's window? Is it pasting text into the program? If a crash is completely random, you'll have a hard time developing a workaround. If specific conditions trigger the crash, you'll be better able to avoid it, but only if you know what those conditions are. Once you notice a pattern, try to deliberately trigger the problem, testing variations on the triggering pattern so you can observe the results.
- ✓ **Try alternative activation methods.** Many programs provide multiple ways to activate a feature, such as menu items, icons in a button bar, and keyboard shortcuts. It's conceivable, but unlikely, that one of these methods will work, whereas others won't work.
- ✓ **Try similar features.** Sometimes you can work around a problem by using a feature that's similar to the one you want to use. For instance, you

might be able to save a document with a file export option rather than the usual save option. Even if the results aren't optimal, they might be good enough for a temporary workaround.

- ✓ **Try another environment.** Bugs are sometimes related to the environment in which the program runs, such as the desktop environment, the X server's color depth, the fonts being used, or even the Linux kernel version you're running. You can try changing such factors to see if they have any effect.
- ✓ **Try another program.** If the bug is one of those stubborn ones that you can't seem to get rid of no matter what industrial-strength solutions you try, you should simply ditch the offending program, at least until the bug is fixed.
- ✓ **Look for help.** Don't be like the stereotypical male who's lost and refuses to ask for directions. Friends, co-workers, or neighbors might be able to offer advice. In the online world, Usenet newsgroups and Web forums can be good sources of information. You may turn up information in a Web search that can help you or find information in the program's own documentation.

Ultimately, working around program crashes and other bugs is more of an art than a science. Every program is unique, and every program's bugs are unique. Thus, you'll have to put in some time to figure out the conditions that trigger the problem and find ways around it.

Reporting Bugs

In the long term, the most important thing you can do about buggy software is to report the bugs. Even if the bug you've found seems obvious, you should report it. The reason is that bugs frequently manifest only under certain circumstances. For instance, a bug might cause a program crash when you save files on your computer, which of course seems like a problem that's hard to miss. This problem might only occur, though, when files are saved to a directory with a hyphen in a directory name. If your home directory happens to have such a name, the problem will always occur for you but it might never occur for one of the developers. Even if you don't notice this connection, reporting the bug will alert the developers to the fact that there is a problem. They might find it upon reexamining the code, or they might ask you to perform some extra steps to help isolate the problem. Either way, this problem won't be fixed unless you report it. (Then again, maybe somebody else will report it—but you shouldn't count on that.)

How do you report bugs? That depends on the program. You should go to the program's home page for directions. You might find the home page listed in an "about the program" option in a menu or listed in the man page or other documentation for the program. As a last resort, try performing a Web search on the program's name. Once you've found the program's home page, look for directions on reporting bugs. Small programs that are handled by just one or two people frequently have simple e-mail reporting procedures—you e-mail the developer with information about the problem. Larger programs with large development teams frequently use Web-based bug-reporting tools. You might need to register with these sites and then fill out a form with information like the program's version number, your distribution, and so on.

If you're lucky, the developer will be able to fix the bug quickly. I've had fixes delivered to me within minutes of reporting a problem. If you're unlucky, you may wait weeks or months for action. In some cases, you may need to recompile the program yourself to get the fix but most developers will make precompiled binaries available to you. You can then install them using the package tools described in Chapter 6. These fixed versions might take weeks or months to work their way into the official distribution archives, though, so you should keep an eye on the package to ensure that it's not upgraded by distribution upgrade tools until you're sure that the fix has been incorporated into the official distribution copy.

Improve Printing Performance

Most Linux performance issues relate to speed or reliability of the system as a whole or of specific programs. An area that straddles the line, though, is printing. Linux's printing subsystem is unusual compared to that of Windows, and these differences can cause performance problems, both in terms of speed and in terms of the quality of the output. Being aware of the way the system is structured can help you improve matters. Two areas that are particularly important are picking the correct drivers and setting an appropriate printer resolution. If you're in the market for a new printer, knowing how Linux interacts with the printer will also help you select a model that will work well.

Understanding Linux Printing

If you've used Linux for long, you've probably seen the print dialog boxes, such as the Mozilla Firefox print dialog box shown in Figure 7-2. On the surface, these dialog boxes seem similar to those used by Windows or Mac OS. Beneath the surface, though, lurks an entirely different sort of printing beast.

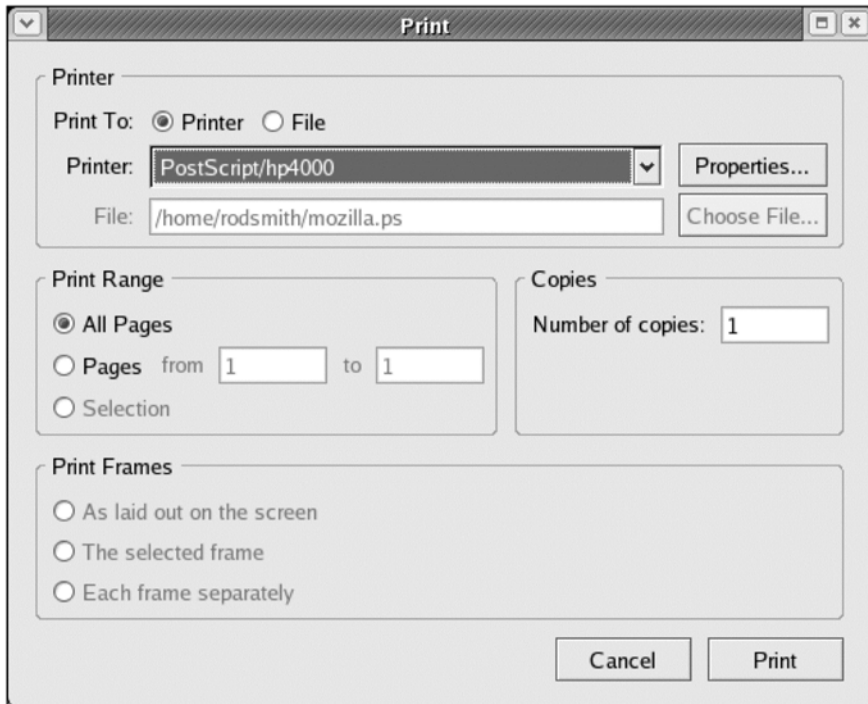


Figure 7-2

Most GUI Linux applications present print dialog boxes that are similar to those used in Windows or Mac OS.

Getting the most from Linux printing means understanding that beast. Indeed, sometimes a printer refuses to work at all when you initially install it; you'll feed it a print job, only to have nothing emerge from your printer.

The Linux community is engaged in a shift from one printing paradigm to another one. The original Linux printing system is lifted entirely from Unix, as implemented in the 1980s. This system uses one of two sets of software: the Berkeley Standard Distribution Line Printer Daemon (BSD LPD) or the next-generation LPRng system. (For simplicity, I refer to both as LPD printing systems.) Both tools provide simple one-way printing paths—applications pass data to the printing daemon, which sends it on to the printer. Applications have no way of knowing anything about the printer, so if you need to use an unusual printer feature (such as a capacity to use extra-wide paper), the application must enable you to enter that information manually.

By 2005, most Linux distributions had switched to a new printing package, known as the Common Unix Printing System (CUPS; <http://www.cups.org>). CUPS provides a compatibility mode that works just like

the LPD system, but it also does more. Most importantly, CUPS provides feedback to the application. A program that uses CUPS can ask CUPS about a printer's capabilities and receive that information. The application can then provide appropriate options to the user, such as wider margins if the printer supports them.

Of course, in order to deliver information about the printer to the application, CUPS must know something about the printer. This is part of a wider issue: Any printing system, on any OS, must know how to communicate with a printer. Unfortunately, printers have different capabilities and use different printer languages. That's why printers come with CD-ROMs that hold Windows drivers—when you use Windows, you install the drivers so that Windows can “talk” to the printer. Without the driver, you can't use the printer.

The Unix tradition has held that printers are either *line printers* or *PostScript* printers. Line printers are simple but fast printers that print text without special formatting, but that can't handle graphics, multiple fonts, or other modern printing staples. PostScript is a computer language that's optimized for generating printed output. PostScript printers understand PostScript, so if you send a PostScript file to a PostScript printer, the result is printed output. This assumption greatly simplifies an LPD printing system; applications can just produce raw text or PostScript output and feed it to the printing system, which can then pass the result to the printer without further processing.

Unfortunately, most printers connected to Linux systems aren't PostScript models. Thus, both LPD systems and CUPS incorporate another software component to help bridge the gap: Ghostscript (<http://www.cs.wisc.edu/~ghost/>). This program is a PostScript interpreter that resides on a computer rather than in a printer. Ghostscript can produce output in any of dozens of graphics file formats, including both traditional computer graphics formats like JPEG and formats that can be understood by most printers. When you configure the printing system to know what printer model you've got, it can pass the PostScript produced by an application through Ghostscript and send the output to the printer, which in turn produces printed output.

One final detail of Linux printing deserves attention: Both LPD and CUPS are network-enabled printing systems. That is, when you've set up the printing system to work locally, it's relatively simple to configure it to accept remote print jobs or to send print jobs to other systems. In practice, most Linux distributions disable this functionality by default as a security measure, but the functionality is there. Particularly in the case of CUPS, this can greatly simplify network printing configuration. CUPS can automatically detect printers that

are served by other CUPS computers. This feature is particularly important on mid-sized and large networks, which are likely to have many printers, as well as frequent changes in printers. Configure each CUPS computer to work on the network and any changes will propagate automatically.

To sum up, then, a modern Linux system using CUPS provides printing tools to applications that give them basic information on a printer's capabilities. The application produces PostScript output and sends that output to CUPS, which passes the PostScript through Ghostscript to produce data that the printer can understand. CUPS then sends this output to the printer, which in turn produces printed output.

Unfortunately, this sequence isn't foolproof. CUPS might try to send its output to the wrong printer or CUPS might not be configured to use the right driver (that is, CUPS might give the wrong driver information to Ghostscript). Even aside from the possibility of producing reams of gunky output, printing might take too long because of suboptimal driver options.

Degunking Your Printer

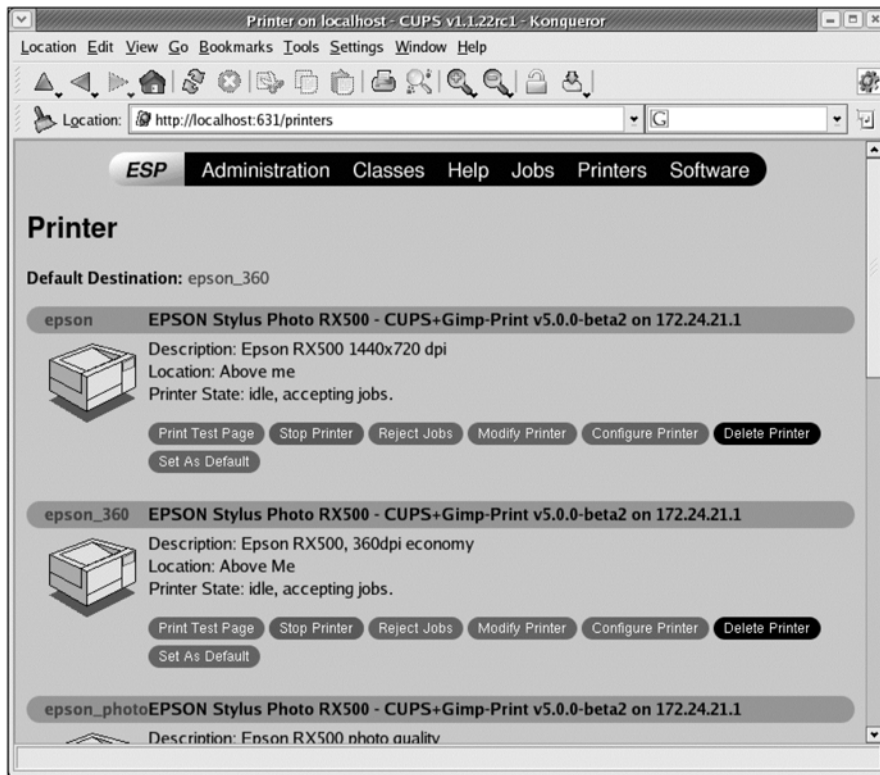
Now that you understand a bit about how the pieces of a Linux printing system fit together, it's time to pull out the cleaning solutions and get it all working smoothly, with no hints of gunk in the mechanism. This process begins with a tour of CUPS configuration tools. You can then check that you're using the right output device, pick the correct driver, set the printer's resolution and other speed-related options, and (if desired) enable network printing.

NOTE: Because CUPS now dominates Linux printing, this description applies only to CUPS. If your system uses BSD LPD or LPRng, you might want to try upgrading to CUPS.

CUPS Configuration Tools

Considered as a network tool, CUPS borrows heavily from the Hypertext Transfer Protocol (HTTP) upon which the Web is based. In fact, CUPS provides a Web-based configuration interface. In a Web browser, type **http://localhost:631** to configure CUPS on the local computer. The result should be a list of tasks, such as Manage Printers and Manage Jobs. Click Manage Printers and the display should change to show all the currently defined printers, as shown in Figure 7-3. (If you've not yet configured any printers, this list will be empty.)

NOTE: Many distributions provide their own tools for administering CUPS. For instance, Fedora and Red Hat provide one called Printer Configuration (aka `system-config-printer`). These tools provide options similar to those of the standard CUPS tool, but interface details differ.

**Figure 7-3**

CUPS provides a Web-based tool for printer administration.

If you've defined a printer but it's not working, you can adjust it by clicking the Modify Printer or Configure Printer options in the printer's area. The Modify Printer option lets you change the options you set when you defined the printer, such as the printer's name, the output device, and the printer driver to be used. The Configure Printer option lets you fine-tune certain options, such as the printing resolution.

If you haven't yet defined your printer, you can do so by clicking Add Printer. (This option appears after all the existing printers and isn't visible in Figure 7-3.) This action steps you through the printer-creation process. If you haven't already done so, you'll be asked to enter a username and password. These should be root and the root password, respectively. You'll be asked to enter three main classes of information:

- ✓ **Identifying information**—This consists of a name, a location, and a description. Programs use the name to identify the printer, and so it should be a single short and descriptive word. The other information is purely descriptive, so enter whatever you like.
- ✓ **A device**—The computer knows how to locate the printer based on its device type—whether it’s a parallel printer, a USB printer, a network printer, or something else. This selection is described in greater detail shortly, in “Picking the Right Output Device.”
- ✓ **A make and model**—These collectively determine the driver that CUPS uses to print to the printer. An incorrect selection will result in gibberish being printed. This topic is covered in more detail shortly, in “Picking the Right Driver.”

Once you’ve defined a printer, it will appear in the printer list. You can then fine-tune it by clicking the Configure Printer link. This will let you set the printer’s resolution, default paper size, and so on.

Picking the Right Output Device

Before you can print, you must know how to identify your printer. In computer terms, this means the output device. Most printers sold today have USB interfaces, so you’d specify a USB port. (More on this shortly.) Some printers, including most printers sold through the 1990s, have parallel ports, which connect to the parallel port on the back of your computer via a parallel printer cable. A few older printers use RS-232 serial ports or more exotic interfaces. CUPS also lets you select a network printer instead of a local printer device, but the details of this selection depend on the network protocol.

You pick the output device using the CUPS device setting. This consists of a list of predefined devices. For parallel and RS-232 serial printers, you need only select the port. Most computers have just one parallel port, so Parallel Port #1 is the appropriate selection for them. If you have an old RS-232 serial printer, pick Serial Port #1 or Serial Port #2, depending on the port to which it’s connected. (In either case, if you’ve added extra ports via an add-in card, you may need to select a higher number.)

TIP: If you’re not sure what port your printer uses, try sending a text file to the raw devices associated with various ports. For instance, typing `cat sample.txt > /dev/lp0` sends the file to the first parallel port (`/dev/lp0`), and `cat sample.txt > /dev/ttyS1` sends the file to the second RS-232 serial port (`/dev/ttyS1`). If you send a file in a format that the printer can understand, the file should print. If not, you should at least see a blinking light or message of some sort on the printer’s control panel, verifying that it’s receiving the input. USB printers usually don’t have device files associated with them, though, so this trick won’t work with them.

For USB printers, you should pick one of the USB printer options; however, these options change dynamically—they should include the name of the printer, as in USB Printer #1 (EPSON Stylus Photo RX500). If you’ve got just one USB printer, it should be identified as the first one. If you see USB printer options but none includes a printer model, chances are this means the printer was turned off or was not plugged in when CUPS started. To configure the printer correctly, follow these steps:

Turn on the printer.

1. At a Linux command prompt, type **/etc/init.d/cups restart** as root. This will restart CUPS, causing it to detect the USB printers anew. (You may need to change `init.d` to `rc.d` on some distributions.)
2. In your Web browser, start again from **http://localhost:631** and begin defining or redefining your printer.

Once you’ve done this, you should see your USB printer make and model as one of the USB device options when you attempt to create a new printer definition or redefine an old one.

TIP: *Because CUPS wants to see your USB printers whenever it starts, and because CUPS starts automatically whenever the computer boots, you should ensure that your printer is plugged in and powered on before you start your computer. If you forget, you’ll need to either reboot or perform steps 1 and 2 to have CUPS check again for the printer before you’ll be able to print.*

If you want to use a printer that’s connected to another computer, you can do so. If the printer is connected to another system that runs CUPS, you should skip ahead to “Enabling Network Printing.” Once that’s set up, CUPS should automatically detect other network printers. If the printer is hosted by a Linux or Unix system that runs LPD or by a Windows system, though, you’ll need to select a network printing protocol. In particular, the LPD/LPR Host or Printer option is for LPD printers, and Windows Printer via SAMBA handles Windows printers.

For both LPD and Windows printers, CUPS presents a text-entry dialog box in which you enter the universal resource identifier (URI) for the printer. This consists of a protocol code (`lpd://` or `smb://` for LPD or Windows printers, respectively), a hostname, and a printer name. For instance, to print to the canon printer on the gutenburg computer using LPD, you would enter:

lpd://gutenberg/canon

Many Windows printer shares require use of a username and password. You can enter this information before the hostname:

```
smb://passwd:user@GUTENBERG/CANON
```

This sends `passwd` as the password and `user` as the username. You should change this information as necessary, of course. If a computer requires a password but no username, you can send any dummy string as the username.

Picking the Right Driver

A good part of the trick to configuring printing in Linux is picking the correct driver. Unfortunately, few printer manufacturers explicitly support Linux, and those that do usually do so only through files you must download from their Web sites. (They often hide the files quite effectively, too.) Fortunately, most printers are well supported via free drivers that come with Linux.

When you get to the part of the CUPS configuration that asks for drivers, you'll see lists of printer makes followed by printer models for the make you've selected. If you don't see your printer make, you should do one of two things:

- ✓ **Download more drivers.** If your printer manufacturer is one of the "big names" (Hewlett-Packard, Lexmark, Epson, Canon, and so on), a missing name most probably means that your system is lacking a printer driver package. Check your installation media for packages called `foomatic`, `gimp-print`, or `printer-drivers` and install any or all of these packages that you find. You should then restart CUPS, as described earlier, and begin again.
- ✓ **Select a compatible model.** If your printer is an obscure brand, you may need to select a compatible model. Many laser printers work as Hewlett-Packard or PostScript models, for instance. Consult your printer's manual to learn what model it emulates.

Some printer makes have multiple entries in the listings under different names (HP and Hewlett-Packard, for instance). If this is true for your printer make, select one and proceed. If you don't see your model in the list, go back and try the other entry.

Sometimes, particularly for very new models, you won't see your specific model listed, even if your make is listed. If this is the case, you may need to download and install a more recent printer driver package, or your printer may work with the driver for another model. If you have trouble with this, you may want to consult the Linux Printing Web site, <http://www.linuxprinting.org>. You

can look up your printer make and model to learn what drivers work best with it. You'll find pointers there to driver packages, installation tips, and more.

Once you're done configuring or reconfiguring the printer, you can click the Print Test Page link in your printer's area on the CUPS configuration page. The result should be a one-page test document. If gibberish emerges, chances are you picked the wrong printer model. Go back and review your selection. Note that some models are very similar in name to models that are very dissimilar in operation, so pay very careful attention to this detail. If no output appears, you might have selected the wrong printer model, or you might have picked the wrong output port. If your printer has status lights or an LCD control panel, check them when you print. If you see signs of activity, chances are you picked the wrong driver and the printer is simply rejecting the job. If you don't see any activity, chances are you picked the wrong output device.

Setting the Resolution and Other Printer Options

Once a printer is up and producing legible output, you can tweak its configuration. Some of these settings can dramatically improve (or degrade!) a printer's performance. To make these changes, click the Configure Printer link in the printer's area on the CUPS Web page. The result should resemble Figure 7-4, although many of the details vary from one printer to another.

For the most part, you should leave the default settings in place, particularly if you don't understand them. One that's particularly likely to affect both print speed and quality, though, is Resolution. (For some printers, an option called Print Quality can also affect the resolution, but descriptions such as High and Standard are used for Print Quality, whereas numeric resolution measures, such as 1440x720 dpi, are used for Resolution.) As a general rule, the higher the resolution, the better the print quality but the longer the printout will take to print. The best trade-off point varies with the printer and your own standards for quality and patience. You may want to try several settings; change the option and then produce a test printout. Time the process and judge the quality for yourself.

Another option you may want to check is Media Size. In the U.S., chances are this should be set to Letter, but it defaults to A4 (a European paper size) for many printers. Although you'll be able to print with the wrong media size, your margins will be peculiar.

You can, of course, peruse additional options. Perhaps you'll find some you know you'll want to adjust, such as paper tray options or default duplex settings. You may also want to return to this page after a while. Many drivers provide

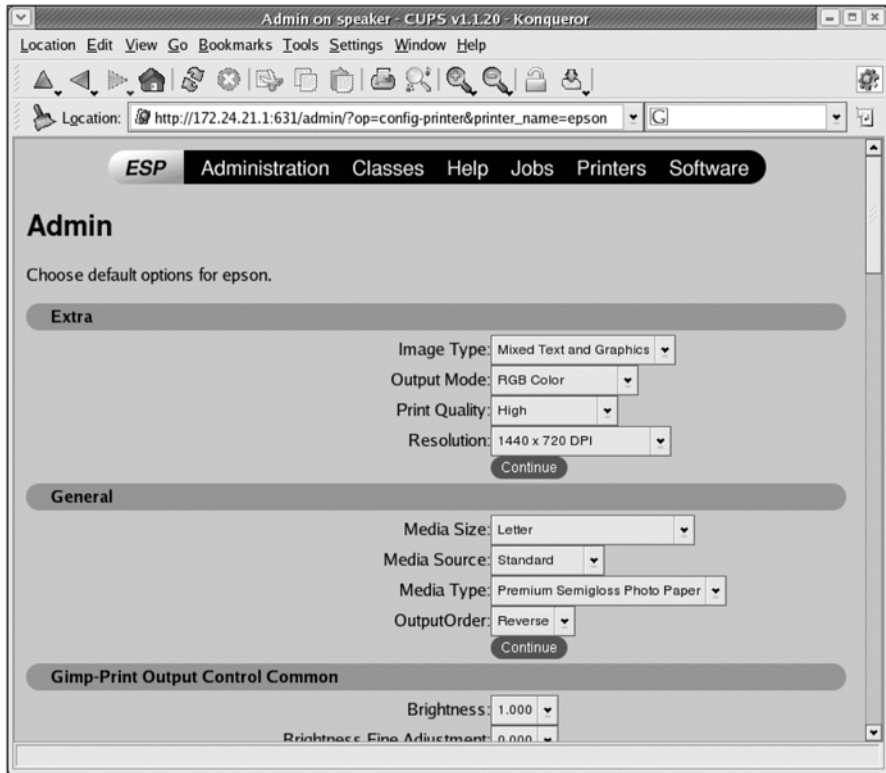


Figure 7-4

You can modify printer resolution and other options after you define a printer.

brightness, contrast, and other settings that can be handy for tweaking the appearance of graphics, for instance.

When you're done making any changes, click one of the Continue buttons. This action saves the changes, whereupon you can test them with a test printout, either from CUPS or from an application.

TIP: The default printer resolution is set when you install the printer. Some CUPS-aware applications provide a way for you to adjust the resolution, but programs that were written with LPD in mind won't be able to do this, so they'll be able to print at only one resolution. If you want to provide an easy way to print at whatever resolution you like, you can create multiple print queues—that is, define a printer multiple times under different names. For instance, you might define queues called `epson_360`, `epson_720`, and `epson_1440` to print at 360 dpi, 720 dpi, and 1440 dpi, respectively. Configure each queue with the resolution and other options desired and you're set.

Enabling Network Printing

One of the great advantages of CUPS is that it greatly simplifies network printing. Rather than define every available network share on every desktop system, you need only configure the desktop systems to support

browsing—the CUPS feature that provides automatic discovery of remote CUPS print servers. The print servers must also be configured to support browsing. Once this is done, changes on the print servers are automatically propagated to clients, so if you add or remove a printer, you needn't adjust the clients. This feature is most useful on mid-sized and large networks, but it can be useful in small offices and even homes.

To use CUPS browsing on a client (that is, a desktop system that you want to be able to access remote printers), you should edit the `/etc/cups/cupsd.conf` file. In particular, look for a line that defines the `Browsing` value and set it to `On`:

```
Browsing On
```

This is the default value in a standard CUPS installation, but some distributions set it to `Off` as a security measure. If you find one instance of this option, continue searching for more; occasionally, it's set one way in one part of the file but then overridden later. You should delete or comment out all but one instance of this option. A few other options can affect the browsing process as well:

```
BrowseProtocols cups
BrowseOrder Deny,Allow
BrowseAllow from @LOCAL
```

Chances are you won't need to adjust any of these options. The first specifies that CUPS is to use its standard browsing protocol, and the second defines whether allow or deny directives are to take precedence. The `BrowseAllow` option is particularly important because it defines the networks with which CUPS communicates when browsing. `@LOCAL` refers to the local network, which is usually correct for a small (or even many mid-sized) networks. If your network spans multiple subnets, you may need to include several `BrowseAllow` directives, each specifying a different subnet. (If this last sentence sounds like Greek to you, it probably means that you don't need to worry about it.)

On the CUPS server (that is, the computer to which the printer is actually connected), you should set the `Browsing On` parameter, as just described. In addition, you should set the `BrowseAddress` option:

BrowseAddress @LOCAL

If this line isn't already present, place it at the end of the file. As with `BrowseAllow` on the client, you may need to add more such lines specifying multiple subnet addresses if your network is particularly large. Also as with `BrowseAllow`, if that sentence makes no sense to you, don't worry about it. Chances are, it doesn't apply to you.

Once you've made these changes, restart CUPS on both the client and the server by typing `/etc/init.d/cups restart` as root. (You may need to change `init.d` to `rc.d` on some systems.) After a few seconds, or a few minutes at most, all the network printers defined on your CUPS server system should become available on the CUPS client system. The printers will appear in the CUPS Web configuration screen (Figure 7-3), and the printers will become available in applications' print dialog boxes (Figure 7-2). You might need to restart your applications before the new printers show up, though.

Picking a Printer for Linux

Linux is a very capable OS, but one of its problems is that not all hardware works with it. Although most printers work well with Linux, this isn't true of all of them. Some printers are simply so new that the necessary drivers have yet to be written. Other printers use proprietary languages and protocols, and their manufacturers aren't very helpful when it comes to providing documentation to Linux developers. In either case, the result is the same: a printer that won't print, or that prints poorly.

If you're unlucky enough to have such a printer, you may need to replace it. If you don't yet have a printer, or if you want to replace one that you've got, you should consider Linux compatibility when buying a printer. Fortunately, a resource exists to help you in this quest: the Linux Printing Web site, <http://www.linuxprinting.org>. This site provides pointers to all sorts of information on Linux printing. In terms of buying a new printer, the best part of this site is its printer database. From the main page, select the Printer Listings link. The result is a page in which you can select the make and model of a printer you're considering. When you've finished, click Show to see the information on that model. (Alternatively, you can select a manufacturer alone to see a list of all the models made by that company, then click a model name to see the information on that model.)

The Linux Printing Web site categorizes printers as falling into four categories:

- ✓ **Perfectly**—Every printer feature works via at least one open source Linux driver package. In the case of multifunction printers, the non-printer functions must also work in Linux.
- ✓ **Mostly**—Most printer features work, or all features work but some produce slightly odd results, such as color that's not quite perfect.
- ✓ **Partially**—The printer provides basic functionality, but important features don't work. For instance, a color printer might print only in black and white.
- ✓ **Paperweight**—The printer doesn't work at all. You can't get even a basic printout from it using open source Linux tools.

In my experience, these ratings are somewhat pessimistic. For instance, I own an Epson Stylus Photo RX500, which the Linux Printing Web site lists as being Partially supported; however, I'd classify it as working Mostly, based on the descriptions on the Linux Printing Web site. I did have to jump through a few hoops to get it working, though. Most notably, I had to install a beta-level version of the GIMP-Print drivers. This isn't the sort of thing I'd recommend for novices. Once those drivers become more stable, perhaps the listing for the Epson RX500 will be upgraded to Mostly supported.

Overall, I'd recommend you stick with printers that are listed as Mostly or Perfectly supported. If you spot a printer that's listed as Partially supported or even one that's listed as a Paperweight, you should avoid it unless the description indicates that it can be made to work by following some specific procedure. Even then, you should be wary.

How should you search for a printer? The Linux Printing database contains entries for a huge number of models, including many that have been long discontinued. Thus, your best approach is probably to check your local computer stores and mail-order outlets for desirable models. Take notes and settle on half a dozen or so models that will suit your needs. Once this is done, look up those models in the database. Some are likely to be poorly supported, so you can scratch them off the list. With luck, you'll be left with at least one printer that will work well. If not, go back to the stores and broaden your search.

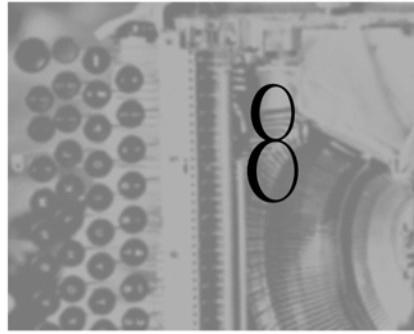
Note that print quality under Linux may not be the same as it is under Windows. Thus, the print samples you see in stores and print quality ratings you see in magazines and on non-Linux Web sites may be meaningless. The reason is that print quality has as much to do with drivers as it does with the hardware. Because the Linux drivers are unrelated to the Windows drivers, print quality

from Linux could be better or worse than print quality from Windows. Indeed, multiple Linux drivers are available for many printers, and one driver might produce much better results than another. If getting the best print quality is important to you, your best bet is to ask other Linux users for advice. Friends, neighbors, co-workers, Usenet newsgroups, Linux-oriented Web forums, and so on can be very helpful resources in this respect. Asking around can also be helpful if you can't seem to find a printer that's supported under Linux and that meets your other requirements. Perhaps a printer, like my Epson RX500, shows up as being poorly supported in Linux but will in fact work better than the database suggests.

Summing Up: Keeping Your System Zippy and Stable

Computer speed and stability are important factors for everybody. Linux has a well-deserved reputation as a stable platform, but that doesn't mean it will always perform smoothly. Program bugs, bloated software, insufficient hardware, and other problems can slow down the system, cause individual programs to crash, or even cause the entire computer to crash. Many small factors can improve matters, from using more reliable or smaller programs to disabling unnecessary features in the programs you do use. Still, chances are you will occasionally encounter programs that crash. When you do, fixing them may be difficult or impossible if you're not a programmer, but you may be able to work around the problem, and if the program is actively supported, you can report the bug so that it will be fixed.

Another area of performance that can be greatly influenced by system configuration is printing. Finding, using, and properly configuring the right drivers can help speed up your printing and produce the best-quality output possible.



Managing Processes

Degunking Checklist:

- ✓ Understand how Linux starts programs automatically when it boots.
- ✓ Change processes that are started during the boot process to improve performance.
- ✓ Locate programs that consume too much CPU time or memory.
- ✓ Terminate programs that have gotten out of control.
- ✓ Tame programs that need to be running but that might interfere with other programs.

A *process* is a running program. At any given moment, a Linux system is likely to support dozens or even hundreds of active processes—your X server, your window manager, a word processor, a printing tool, a program to set the time in the background, and so on. On a properly degunked Linux system, every one of those programs is necessary, or at least desirable; however, processes sometimes run when they shouldn't or run out of control in one way or another. (Chapter 7, “Improving Software Performance,” introduced some potential problems.) Thus, knowing how to manage running processes is important for keeping a system degunked. If you just let everything run as it will, sooner or later your system will begin behaving erratically and you'll think you have to reboot it. Fortunately, Linux is smarter than that: Linux provides tools to locate and terminate misbehaving processes or to reign them in without killing them. First, you should understand a bit about how Linux boots and begins running the bare essential programs that make up a Linux system. This will enable you to change the software base upon which everything else is built.

Understand the Linux Startup Procedure

When you boot a typical Linux system, you see a variety of messages on the screen, followed by a login prompt. Depending on your distribution and settings, many of the messages may relate to programs being launched. Thus, paying attention during this process can help you track down some gunk—if you notice a message about an FTP server being launched but you don't want to run an FTP server, you know you should go looking for that FTP server to disable or remove it. Whatever the form of the startup messages, you can also intervene by reconfiguring the startup scripts to do what you want them to do.

The Tangled Web of Startup

The Linux boot process involves several steps, which are outlined in Figure 8-1. This figure presents a somewhat simplified view of things, but it's enough to get across some key points.

The Linux boot process begins with the computer's Basic Input/Output System (BIOS), which reads the *boot sector* from a disk. This boot sector contains a *boot loader*, which is a small program that takes over the boot process and loads another boot loader (a *secondary boot loader*) or an OS kernel, such as the Linux kernel. Linux distributions for the x86 CPU line ship with the Grand Unified Boot Loader (GRUB) or the Linux Loader (LILO) boot loader. Both are capable of reading the main Linux kernel file from disk and starting it running.

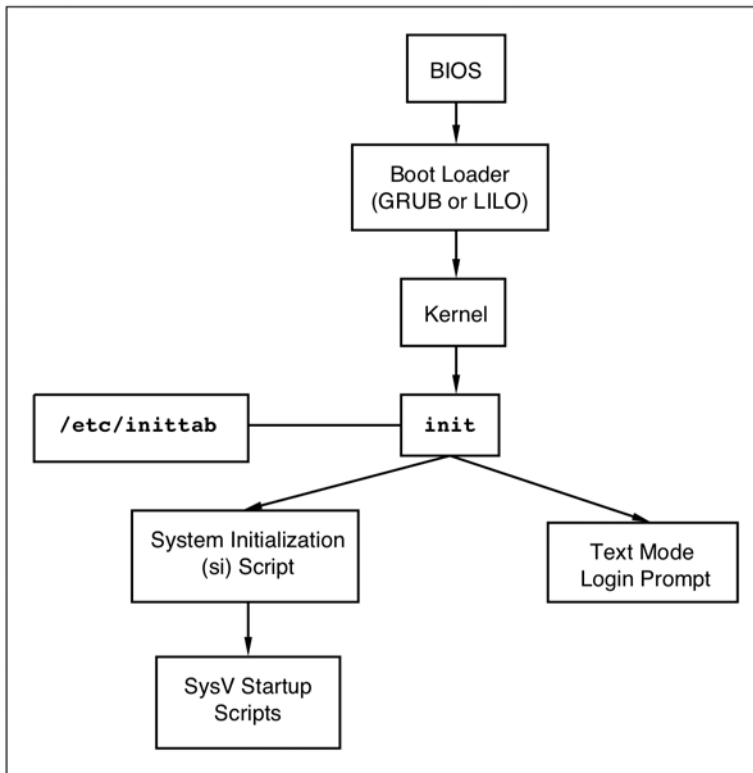


Figure 8-1

The Linux boot process involves several distinct stages, each of which handles particular tasks.

NOTE: The boot loader is typically configured during Linux installation. You can change the configuration after the fact if necessary. This is most commonly done when upgrading the Linux kernel—the boot loader must be told about the new kernel file.

Once the Linux kernel is loaded, it begins looking for hardware and performing other system checks. On a traditional Linux system, this stage of the boot process is marked by messages scrolling past on the screen:

```

Linux version 2.6.8.1 (rodsmith@speaker) (gcc version 3.3.3 (SuSE
Linux)) #4 Sun Oct 17 10:58:55 EDT 2004
BIOS-provided physical RAM map:
BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
BIOS-e820: 000000000009fc00 - 00000000000a0000 (reserved)
BIOS-e820: 00000000000f0000 - 0000000000100000 (reserved)
BIOS-e820: 0000000000100000 - 0000000017ff0000 (usable)
  
```



```

BIOS-e820: 0000000017ff0000 - 0000000017ff8000 (ACPI data)
BIOS-e820: 0000000017ff8000 - 0000000018000000 (ACPI NVS)
BIOS-e820: 00000000ffff0000 - 00000000100000000 (reserved)
383MB LOWMEM available.
On node 0 totalpages: 98288
  DMA zone: 4096 pages, LIFO batch:1
  Normal zone: 94192 pages, LIFO batch:16
  HighMem zone: 0 pages, LIFO batch:1
DMI 2.3 present.

```

These messages continue on for quite a while, and most of them are highly system-specific. They contain information that's sometimes helpful in diagnosing hardware problems. Some modern distributions replace this listing early on with a GUI display that's less intimidating to Linux newcomers but that provides less useful diagnostic information.

TIP: *If you want to see your kernel boot messages after the system has booted, type `dmesg | less` at a Linux command prompt soon after booting. This command displays the boot messages and passes them through the `less` pager, which enables you to see the messages a screen at a time.*

Once Linux has finished its hardware checks, it launches a program called `init`. In Linux, every program is launched by another program. The program that does the launching is called the *parent*, and the launched program is called the *child*. As the first regular program (launched by the kernel itself), `init` is special. It launches additional programs in the boot process. It also “adopts” programs that become “orphaned” because their parents terminate before they do. In any event, `init` is controlled through the `/etc/inittab` file. Like most Linux configuration files, `/etc/inittab` is a plain text file, so you can read it in a text editor or use `less` to view it. (Type `less /etc/inittab` to do so.) Its exact format is unimportant for the moment, but the line beginning `si` specifies a script that continues the boot process. The name of this script varies from one distribution to another, though. For instance, in Fedora and Red Hat, it's `rc.sysinit`:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

In SuSE Linux, it's `boot`:

```
si::bootwait:/etc/init.d/boot
```

No matter what its name, the system initialization script ends up performing a number of important startup tasks, including running a series of *startup scripts*—scripts that launch specific programs or perform specific tasks. In most distributions, these startup scripts take the form of *System V* (*SysV* for short) startup scripts, which are located in a directory called `/etc/rc.d`, `/etc/rc.d/init.d`, or `/etc/init.d`. Subdirectories in one of these directories have filenames of the form `rc?.d`, where `?` is a number from 0 to 6. These numbers refer to *runlevels*, which are numbered sets of programs that should be run. Runlevels 0, 1, and 6 are special, but intervening runlevels can define sets of programs that can serve particular functions, such as runlevels with and without active networking features. The `/etc/inittab` file defines the runlevel that Linux uses when it first boots:

```
id:5:initdefault:
```

This line sets the default runlevel to 5, which is typical. Many distributions define runlevel 3 as a basic text-mode boot and runlevel 5 as a boot to a GUI login system. (Debian and Gentoo are exceptions to this rule; they attempt to run X at all runlevels between 2 and 5 but fall back to a text-mode login if X won't start.)

Mainstream Linux distributions include literally dozens of SysV startup scripts. These scripts are very useful in controlling what software runs by default on a Linux system—by reconfiguring these scripts, you can shut down unnecessary software, start software that's not running by default but that should be, and otherwise modify the system.

NOTE: *Slackware is the only mainstream Linux distribution that does not use SysV startup scripts. Instead, for each runlevel it provides one script to start all of the programs that should be active in that runlevel.*

Once a system is running, of course, users can launch their own processes. This happens only through user logins, which are mediated by local text-mode, local GUI, or remote login programs. The main text-mode login program (appropriately enough called `login`) is launched directly by `init`. Some distributions also use `init` to launch a GUI login process, but others use SysV startup scripts to handle this task. Most network login protocols are launched through SysV startup scripts or through a *super server*, which is a server that stands in for other servers. (Super servers are launched through—you guessed it—SysV startup scripts.) Thus, user processes, like everything else, trace their way back to `init`.

Intervening in the Startup Process

Sometimes it's necessary to change what gets started during the boot process. These changes don't alter the broad outline of the startup process, which is illustrated in Figure 8-1; instead, they change details such as which SysV startup scripts run. Because SysV startup scripts start servers and other important system components, you can alter how Linux behaves by changing these scripts. Most distributions also provide a special startup script, known as a *local startup script*, that's intended to be edited directly to launch programs you want to have running but for which no SysV startup script exists. You can use such a script to help customize your system by running unusual local programs.

NOTE: You don't normally use any of these startup scripts to launch user programs, such as desktop applets. Rather, SysV and other system startup scripts run low-level system utilities such as network initialization tools, servers, and login programs. To run a program as a user whenever you log in, check your desktop environment configuration or user login dot files.

Configuring SysV Startup Scripts

As noted earlier, SysV startup scripts are usually located in `/etc/init.d`, `/etc/init.d/rc.d`, or `/etc/rc.d`. (Sometimes distributions provide links so that two or all three of these locations will work.) You might not want a SysV startup script to run all the time, though. You can, of course, uninstall a package if you never want it to run, as described in Chapter 6, "Cleaning Out Unused Packages"; however, this may be overkill. Perhaps you want to continue using the program but only launch it at certain times, or perhaps you want to run the program via a super server rather than via a SysV startup script.

NOTE: Not all programs have SysV startup scripts. Because these scripts are used to launch servers and other low-level system services, only packages for such programs provide SysV startup scripts. Most user programs, such as text editors, compilers, and e-mail readers, aren't started through SysV startup scripts.

This is where runlevels come in. Each runlevel starts a specified set of SysV startup scripts. To define runlevels, distributions use subdirectories of `/etc` or of the main SysV startup script directory. These subdirectory names take the form `rc?.d`, where `?` is the runlevel number, from 0 to 6. Each runlevel directory contains links to SysV startup scripts, but the links have altered names, of the form `S##script` or `K##script`. In both cases, `##` is a two-digit number and `script` is the original SysV startup script name. If the link's name begins with `S`, Linux passes the `start` option to the script, which causes the script to start the service it controls; if the link's name begins with `K`, Linux passes the `stop` option to the script, which causes the script to stop (kill) the service it controls.

The two-digit number serves as a sequence number; services are stopped and started in order based on those numbers.

This set of features enables you to control what processes are started and stopped in each runlevel. You can create or rename links to add a service or change whether it's active or inactive in a runlevel. The main problem with this approach is the sequence numbers. Some services are dependent on others being active to start or should not be stopped before others. For instance, network servers all rely on basic networking features. In most distributions, the network is activated through a SysV startup script called `network` or `networking`, so that script must have a startup sequence number that's lower than the startup sequence number for any network server. The opposite is true for the shutdown scripts. Furthermore, startup sequence numbers aren't standardized across distributions; a service with a number of 40 in one distribution might have a number of 55 in another. For these reasons, you shouldn't try manually adjusting SysV script links for particular runlevels. Unless you have an intimate understanding of the sequence for your distribution and of the needs of a particular service, you're likely to get it wrong and cause the service to not start or stop correctly, or to interfere with other services.

NOTE: *The Gentoo distribution is a bit odd. Rather than use numbered runlevels, Gentoo uses an arbitrary number of **named** runlevels. Furthermore, Gentoo's SysV startup scripts contain dependency information, obviating the need for startup sequence numbers—if a script is run but some depended-upon service is not yet running, Gentoo simply runs the startup script for that service before proceeding.*

In practice, managing the SysV scripts that run in specific runlevels is a task that's best handled via special tools. Some of these tools are present in several distributions, but others are highly distribution-specific:

- ✓ **chkconfig**—This program is a fairly basic command-line tool that's present in most RPM-based distributions, including Fedora, Mandrake, Red Hat, and SuSE.
- ✓ **rc-update**—This program is the logical equivalent of `chkconfig` for Gentoo systems, but the details of its operation differ.
- ✓ **ntsysv**—This program is an interactive text-mode program for changing the services that are running in a given runlevel. It's available for Fedora, Mandrake, and Red Hat.
- ✓ **sysv-rc-conf**—This program is similar to both `chkconfig` and `ntsysv` in that it provides both command-line and interactive text-mode controls of SysV startup scripts. It's commonly used on Debian systems.

- ✓ **Service Configuration**—This tool, aka `system-config-services`, is a Fedora and Red Hat GUI SysV startup script handler.
- ✓ **YaST and YaST2**—These are the text-mode and GUI configuration tools of SuSE, and they provide the means to edit runlevels, among many other things.
- ✓ **ksysv**—This is a GUI runlevel editor that's similar in broad strokes to `ntsysv` and that's part of KDE (it's usually distributed in the `kdeadmin` or `ksysv` package). It can be used with many popular Linux distributions but not with Gentoo or Slackware. Also, although it can be used with Fedora, it's not included in Fedora's KDE packages. The first time you run `ksysv`, you'll be asked which distribution you're using so that `ksysv` can adjust itself to that distribution's file locations and other features. Unfortunately, `ksysv` doesn't automatically handle startup script sequence numbers, which makes it unsuitable for use by those unfamiliar with this detail.

Because there are so many tools for managing SysV startup scripts, I can't describe them all here without causing you severe somnolence. Thus, I focus on just one: Fedora and Red Hat's Service Configuration tool. YaST, YaST2, `ntsysv`, and `sysv-rc-conf` in its interactive mode are similar, although most of these tools use a purely text-based display. To launch the program, type **system-config-services** at a GUI command prompt or select the tool from your desktop environment's main menu. You'll then see a display similar to the one shown in Figure 8-2.

The Service Configuration window shows lists of services on the left, with a description and status information in panes on the right. Just above these panes is a line that describes the runlevel—both the current system runlevel and the runlevel being edited. (Both are at runlevel 5 in Figure 8-2.)

To permanently enable or disable a service for a given runlevel, check or uncheck the box next to the service name. This change will make the service start or stop the next time you boot the computer, but it won't immediately enable or disable the service. To do this, click the Start or Stop button in the button bar. You can also click Restart to restart the service. This action shuts down the service and then starts it up again, which is sometimes necessary if you make changes to a server's configuration and want it to immediately implement those changes.

NOTE: *Service Configuration and some other SysV startup script managers also handle servers that are launched via the `xinetd` super server. When using a configuration editor, for the most part, you needn't be concerned with this distinction; however, some options aren't available for `xinetd`-mediated servers. In particular, you can't directly start, stop, or restart such servers. Instead, you should enable or disable the server and then restart `xinetd` itself via its entry in the Service Configuration tool.*

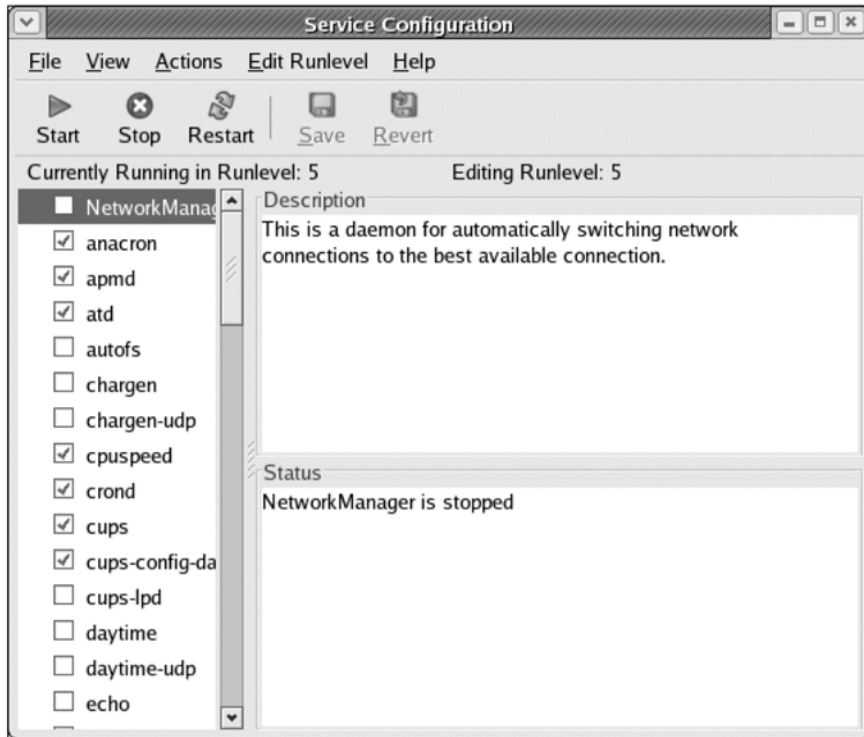


Figure 8-2

SysV script editors present lists of services that run in one or more runlevels, often with descriptions of what each script does.

Linux boots into just one runlevel when you start the computer. You can set this default runlevel by editing the `id` line in `/etc/inittab`. (This line was described earlier, in “The Tangled Web of Startup.”) If you want to edit the services that are active in a given runlevel, you can select the target runlevel from the Edit Runlevel menu in Service Configuration.

To degunk your default runlevel, go through the list of services and read their descriptions. If you don’t understand something, leave it alone. If you come across something that you’re certain you *don’t* want running but that is configured to run by default, disable it. For instance, you might find that the `cups-lpd` service is active. As the description specifies, this is a server that lets the Common Unix Printing System (CUPS) accept print jobs using the old Line Printer Daemon (LPD) protocol. If you know you don’t want to share printers, you can safely disable this feature. (The `cups-lpd` server is handled by `xinetd`, so you’ll need to disable `cups-lpd` and then restart `xinetd`.)

Using Local Startup Scripts

SysV startup scripts are convenient for managing standard services, such as servers and other system tools that ship with a distribution. Sometimes, though, you want to do something that wasn't anticipated by the distribution's maintainer. For instance, perhaps you want to run the Folding@Home program (<http://folding.stanford.edu>), which donates your unused CPU time to a scientific project that investigates the assembly of protein molecules or perhaps you've compiled a standard server yourself rather than install it from a package and you need some way to launch it. Assuming you don't want to manually start such programs yourself, you have two options:

- ✓ You can create your own SysV startup script from scratch or modify an existing one to suit your purposes. This is a suitable option if you understand how such scripts are put together, but if you're unfamiliar with Linux, this task can be quite daunting.
- ✓ You can use a *local startup script*, which is a special startup script designed to handle system-specific tasks. Typically, you'll place a simple call to the program you want to start within this script, causing the program to run whenever the computer is booted.

Of these two options, local startup scripts are definitely the easier to use. Unfortunately, although most Linux distributions support such scripts, their names and locations are not standardized. Some examples include the following:

- ✓ **Debian**—All executable files in `/etc/rc.boot` are treated as local startup scripts, but the Debian developers discourage the use of this tool. These scripts run before most SysV startup scripts.
- ✓ **Fedora, Mandrake, and Red Hat**—The `/etc/rc.d/rc.local` file is the local startup script. It runs after the SysV startup scripts.
- ✓ **Gentoo**—The `/etc/conf.d/local.start` script serves as the local startup script. It's run after the default runlevel's SysV startup scripts.
- ✓ **Slackware**—The `/etc/rc.d/rc.local` file serves as the local startup script. It runs at the end of the boot process.
- ✓ **SuSE**—The SuSE local startup script is called `/etc/init.d/boot.local`. It runs before the SysV scripts for the startup runlevel.

To add a program to one of these scripts, load it into a text editor and add the command you'd type at a command prompt to launch the program to the end of the file. For instance, suppose you've compiled an obscure server, `megaserv`, and placed its binary in `/usr/local/bin`. You might launch it by placing a line like the following in the local startup script file:

```
/usr/local/bin/megaserv
```

You can add any options you like to the call, as well. Most servers can be launched just like this, but some non-server programs and a few servers should be launched with a trailing ampersand (&):

```
/usr/local/bin/megaserv &
```

This symbol tells Linux to launch the program in the background. (Most servers do this automatically, but most non-servers don't.) Without the ampersand, the startup script will stop executing until the program terminates. The result is that subsequent lines in the startup script won't execute, and subsequent steps in the startup process (beyond this startup script) won't work, either.

Programs run from a local startup script (or a SysV startup script, for that matter) are normally run as root. This is fine for many servers and other system programs. Sometimes, though, you want to run a program as an ordinary user. To do so, you can use the `su` command, which switches the user identity. You can use `su` at a command prompt to acquire root privileges or those of another user, or you can use it in a startup script to launch a program as an ordinary user:

```
su -c '/usr/local/bin/megaprogram &' megauser
```

The `-c` option passes a command (in single quotes) to be executed—in this case, `/usr/local/bin/megaprogram`. The command terminates with the name of the user who should run the program, as in `megauser`. Note that the ampersand, if necessary, appears within the single quotes of the command that `su` is to execute, not after the `su` command line as a whole.

Ordinarily, local startup scripts are empty or nearly empty as delivered by distributions. (They may have some comments and perhaps a bit of housekeeping code.) Thus, degunking them shouldn't be necessary, at least not on a freshly installed system. They can be good ways to help customize your system by automatically launching programs that you really do want running, but for which you don't have SysV startup scripts.

Locate Misbehaving Programs

Sometimes programs are like two-year-old children throwing temper tantrums—they just behave badly. Rather than scream and yell and throw toys, though, misbehaving computer programs usually consume too much CPU

time or memory. The result can be overall system performance problems because these resources become unavailable to other programs. Chapter 7 described typical symptoms of this sort of problem and covered some solutions (such as buying more RAM). The rest of this chapter is devoted to tracking down particular misbehaving programs and fixing the problem, at least temporarily.

Finding CPU Hogs

The Linux kernel is in charge of allocating CPU time to every running process. Processes can request CPU time, and if it's available, the kernel grants that request. Most programs are idle most of the time—that is, they don't consume much CPU time. For instance, even when you're furiously typing in an e-mail message, your mail client is likely to be consuming very little CPU time because, by computer standards, the computations necessary to process your keystrokes and display the results on the screen are minor.

You can view the overall CPU load on the computer by typing **uptime** at a command prompt. The result is a list of various system statistics, including the amount of time the computer has been running and three *load averages*, which are measures of the proportion of available CPU time being requested by all running programs combined:

```
$ uptime
```

```
15:29:54 up 24 days, 35 min, 23 users,  load average: 1.73, 2.23,  
1.94
```

In this example, the load averages are 1.73 over the past minute, 2.23 over the past 5 minutes, and 1.94 over the past 15 minutes. Because the load average is expressed as a proportion of available CPU time, you needn't know anything about your computer's CPU speed to interpret the value. A computer with no processes requesting CPU time will have a load average of 0.00, and a load average of 1.00 indicates a single process requesting all the available CPU time (or a combination of more than one process, each requesting less than the available CPU time). Values between 0.00 and 1.00 indicate that the CPU is not being fully utilized. Values above 1.00 indicate that the running programs are requesting more CPU time than is available, meaning that none of these programs will perform optimally.

Ideally, your system should have a load average of 1.0 or less. In practice, though, the load average can rise higher than this value because you're running CPU-intensive programs in the background or because multiple people are using the computer, each placing demands on the CPU. Desktop systems often have

load averages of below 1.0, meaning that there's little competition for CPU time among the running programs—typically, the user interacts with just one program at a time, and that program isn't likely to demand 100 percent of the available CPU time.

NOTE: Replacing the CPU with a faster one might not reduce the load average; multiple CPU-intensive processes will still put a high load on the CPU. A faster CPU will finish the individual tasks faster, though. Likewise, a system with a slow CPU can produce a low CPU load but still seem sluggish because the speed problems aren't related to CPU operations.

If your system seems sluggish and uptime reports a CPU load of above 1.0, it's possible that a process has hung—locked itself in an endless loop, consuming CPU time but doing nothing useful. It could also be that you've got one or more legitimate programs consuming lots of CPU time for legitimate reasons. Either way, a good way to track these programs down is to use the `top` program, which is an interactive text-mode program for tracing system resource use. By default, `top` displays a list of processes ordered according to CPU use, as shown in Figure 8-3. About once every two seconds, this list refreshes itself, reflecting processes that start, stop, or change the amount of CPU time they're demanding.

Review your process list, paying attention to the final column in the `top` output (labeled `COMMAND`). Chances are any CPU hogs will appear among the first few entries. The `%CPU` column shows the percentage of CPU time that a program has consumed during the last time slice. In the case of Figure 8-3, the `FahCore_65.exe` process is consuming 97.4 percent of CPU time, followed by `xvnc` at 1.3 percent. If a process remains high in this measure, it's either legitimately consuming lots of CPU time or it's hung. The `TIME+` column can also be useful; it displays the total amount of CPU time a process has consumed. In the case of Figure 8-3, `FahCore_65.exe` has consumed about 5 minutes of CPU time, whereas `xvnc` has chewed up over 141 minutes of CPU time.

How do you interpret these results? Unfortunately, `top` alone can't identify programs that are consuming inordinate amounts of CPU time; you must bring your knowledge of your system to bear, as well. Figure 8-3 was taken on a system on which I'd just started the `Folding@Home` program (`FahCore_65.exe`). This program legitimately consumes a lot of CPU time, but it does so at a low priority, so it shouldn't interfere with other programs. (The `NI` column in `top`'s output shows process priorities, with positive numbers indicating reduced priorities.) The `xvnc` process is a remote login server, and it had been running for over a day; thus, it's racked up a huge cumulative CPU time.

```

rodsmith@halloprillalar:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

top - 15:53:50 up 21:41, 4 users, load average: 1.04, 0.96, 0.46
Tasks: 91 total, 2 running, 89 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.3% us, 1.0% sy, 97.7% ni, 0.0% id, 0.0% wa, 0.0% hi, 0.0% si
Mem: 515124k total, 509572k used, 5552k free, 124016k buffers
Swap: 465800k total, 0k used, 465800k free, 213416k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 5719 folding   39   19 70652 10m  720  R 97.4   2.1   5:04.86 FahCore_65.exe
3842 nobody    15    0 25924 18m 2972  S  1.3   3.6 141:21.85 Xvnc
3556 root       15    0 59948 10m 3800  S  0.3   2.0  2:07.95 X
3923 rodsmith  16    0 27700 15m 12m  S  0.3   3.1  0:06.22 konsole
   1 root       16    0 1616  612 528  S  0.0   0.1  0:00.93 init
   2 root       34   19    0    0    0  S  0.0   0.0  0:00.95 ksoftirqd/0
   3 root        5  -10    0    0    0  S  0.0   0.0  0:00.00 events/0
   4 root        7  -10    0    0    0  S  0.0   0.0  0:00.00 khelper
  16 root       15  -10    0    0    0  S  0.0   0.0  0:00.00 kacpid
  87 root        5  -10    0    0    0  S  0.0   0.0  0:00.30 kblockd/0
100 root       15    0    0    0    0  S  0.0   0.0  0:00.00 khubb
179 root       20    0    0    0    0  S  0.0   0.0  0:00.00 pdflush
180 root       15    0    0    0    0  S  0.0   0.0  0:00.14 pdflush
182 root       13  -10    0    0    0  S  0.0   0.0  0:00.00 aio/0
181 root       15    0    0    0    0  S  0.0   0.0  0:01.16 kswapd0
263 root       25    0    0    0    0  S  0.0   0.0  0:00.00 kseriod
381 root       15    0    0    0    0  S  0.0   0.0  0:00.20 kjournald
1315 root        6  -10 1500  476 396  S  0.0   0.1  0:00.02 udevd

```

Figure 8-3

The top program orders processes according to the CPU time they're using.

If your system is sluggish and you spot a process that you can't identify or that you know shouldn't be consuming inordinate amounts of CPU time, and if that process is chewing up large amounts of CPU time, it's probably your culprit. If you're lucky, you can match the process to a window on your screen, enabling you to go into that program and shut it down or take other steps to make it run more efficiently. If not, you might need to dig deeper to identify the process. Try looking for a man page for the program; for instance, if you didn't know what the Xvnc process was, you might type **man Xvnc**. This might help you determine what the process is and whether it should really be chewing up so much CPU time. If you've identified the process and want to rein it in but don't have direct access to its user interface (because it's a daemon or because it's not responding), consult the upcoming sections "Kill Stray Processes" and "Adjust Process Priorities" for tips on how to proceed. In any event, you should note the process ID (PID) number in the first column of the top output; you'll need this information to kill the process or adjust its priority. Alternatively, you can use top itself to do the job.

Finding Memory Hogs

Sometimes programs misbehave by consuming a lot of memory. This can be because the programs are designed in such a way that they necessarily chew up memory, because they've been asked to perform some task that requires lots of memory, or because they've got bugs that cause them to request too much memory. In any event, the result is that other programs don't have enough memory. Sometimes the memory hog program itself can't get enough memory for its needs.

Linux, like all modern desktop OSs, supports *swap space*, which is disk space configured to act like memory. Swap space increases flexibility, but it's not as fast as RAM, so when the system digs into swap space to meet its memory needs, the system slows down. In extreme cases, the system will spend most of its time moving data between swap space and RAM, a condition that's known as *thrashing*. When this happens because a program is pigging out at the memory buffet, you should attempt to shut down the offending program. Of course, you must be able to identify that program first.

Once again, *top* is a great tool for the job. When it's first launched, *top* sorts entries according to CPU time used. To change it to sort by memory used, follow these steps:

1. Type **F** (that's an *uppercase* letter F; a lowercase *f* has another effect). The result is a screen that summarizes the fields on which *top* can sort.
2. Type the letter corresponding to the field you want to use to sort *top*'s output. A good option is **n**, for the *%MEM* column—that is, the percentage of memory used. Other memory options exist, but they measure certain subtypes of memory use, such as the data size or the amount of swap space used.
3. Press the Enter key. The result should be the main *top* display, as shown in Figure 8-3, but this display will be sorted by the memory option you selected.

As with spotting CPU hogs, you can look for processes that consume too much memory by going down the list that *top* produces. Is any one process consuming far more memory than others? Is that value growing? (If so, this indicates a worsening demand for memory, either because of a bug or because of runaway data processing.) If you don't know precisely what a suspicious process is, you might want to research it further by looking for a man page entry, but if your system is thrashing badly, the time needed to do this may be inordinate—you might prefer to skip straight to the part where you terminate the offending process.

Finding Processes by Name

Sometimes a process doesn't appear at the top of the `top` listing but you still want to terminate it. If you know the name of the process, one way to find it is to use the `ps` utility, which creates a list of processes. Used without any options, the list is likely to be short because it includes only those processes associated with the shell from which you ran `ps`:

```
$ ps
  PID TTY          TIME CMD
 9066 pts/20        00:00:00 bash
 8468 pts/20        00:00:01 nedit
28641 pts/20        00:00:01 xfig
29024 pts/20        00:00:00 xv
29079 pts/20        00:00:00 ps
```

This listing includes some of the same information as a `top` listing—in fact,—`top` is something like a specialized and real-time version of `ps`. You might find the process you want by looking at a plain `ps` listing; however, `ps` provides a huge number of options that expand the number of processes it shows, expand the information provided, or both. A pair of options I find particularly handy are `a` and `x`, as in `ps ax`. This combination lists all the processes running on the computer and also increases the number of fields in the output. Because a typical Linux system is likely to have dozens or hundreds of running processes, the output of this command will probably be overkill. The answer is to use a Linux *pipe* (described in Chapter 3) to pass the result through `grep`, which is a program that searches its input for a string that you specify:

```
$ ps ax | grep xfig
28641 pts/20    S      0:01 xfig fig08-01.fig
29389 pts/26    S+    0:00 grep xfig
```

This example searches for a process called `xfig`, or any process that includes that string. In this example, two processes are found: the target `xfig` process and the `grep` command to search for `xfig`. As with the earlier example of `ps` use, the PID number is in the first column—it's 28641 for the target `xfig` process. If it had become unresponsive or was consuming too much CPU time, you'd need this number to kill the process or reduce its priority, as described in "Kill Stray Processes" and "Adjust Process Priorities."

WARNING! Sometimes a process name is not unique. For instance, you might be running several instances of `xfig`, in which case they'd all show up in the `ps` output. You might be able to tell which one is the problem process by noting any options that appear in the output, such as `fig08-01.fig` (the filename passed to the program) in the preceding example. Adding the `u` option, as is `ps aux` rather than `ps ax`, adds username information to the output, which might be useful if several users are running the same program.

Kill Stray Processes

Frequently, the only way to deal with a seriously misbehaving process is to kill it. Several tools exist to do this job. In many ways the simplest approach is to use the program's own user interface, but this isn't an option if the program has stopped responding to input or if it's a server or some other tool that lacks a user interface. (You may be able to pass the `stop` option to a server's SysV startup scripts or use a GUI SysV startup script manager to kill a server, though.) If you've identified the misbehaving program using `top`, you can use that same tool to kill the troublemaker. Several GUI tools also exist to help with the job. Another approach is to use the most basic tools available, known (appropriately enough) as `kill` and `killall`.

Killing Processes Using `top` or GUI Tools

The `top` utility can help you identify processes that are running out of control, as described earlier in "Locate Misbehaving Programs." Fortunately, `top` is more than just a gunk identifier; it's a gunk cleaner, too! To use it in this way, follow these steps:

1. Locate the problem process, as described earlier.
2. While still in `top`, press the `k` key in `top`. The program will respond by prompting you, between the summary information at the very top of the display and the process list that occupies the bottom of the display, for the PID to kill.
3. Type the PID number from the first column of the process list for the process you want to terminate. The program responds by asking you for a *signal number*. This is a numeric code that Linux sends to processes as a crude form of communication. Table 8-1 summarizes some common signals and their meanings.
4. Type a signal number and press the Enter key, or simply press the Enter key to accept the default signal number of 15.

5. If the process doesn't disappear from the process list within a few seconds, repeat steps 2–4, but use a stronger signal, such as signal 9 rather than signal 15.

Table 8-1 Common Linux Signal Numbers and Names

Signal Number	Signal Name	Purpose
1	HUP	Indicates that a terminal has closed ("hung up," hence the name). Servers often respond to this signal by re-reading their configuration files.
2	INT	Interrupts a program. The kernel sends this signal when you press Ctrl+C in a running program's terminal.
3	QUIT	Terminates a program and leaves a <i>core dump</i> , which is a file that can be used for debugging purposes.
9	KILL	Kills a program ungracefully, without giving it a chance to save files.
10	USR1	This signal's effect depends on the target program. Consult the program's documentation for details.
12	USR2	This signal's effect depends on the target program. Consult the program's documentation for details.
15	TERM	Terminates a program gracefully. The program can usually save open files and otherwise clean up after itself before quitting.
18	CONT	Continues operation after a STOP signal.
19	STOP	Suspends operation. Similar (but not identical) to pressing Ctrl+Z in a running program's terminal.

NOTE: Table 8-1 is not complete; it only lists some of the most commonly used signals. Signal names are often referred to with SIG before the name shown in Table 8-1, as in SIGKILL rather than KILL. Many of these signals aren't used in terminating processes, but you may want to send them in other contexts, such as getting a server to re-read its configuration files.

For killing misbehaving processes, it's generally best to start with signal 15 (TERM), which is a "polite" way to get the program to terminate. If this action doesn't work, signal 9 (KILL) almost always does the job. If a KILL signal doesn't work, and if the process is a serious problem, you may need to reboot the computer. Occasionally, a process can't be killed but isn't a major threat. For instance, zombie processes (described in Chapter 7) can't be killed directly, but zombies don't consume much in the way of system resources, so you can probably safely ignore them.

NOTE: If you run `top` as an ordinary user, you can terminate your own processes using `top` but not system processes (those owned by `root`) or processes owned by other users. To terminate any system process using `top`, or those owned by arbitrary users, you must run the utility as `root`.

Various GUI tools can be used to terminate unwanted processes. Some of these, such as the KDE Process Monitor (KPM), are basically GUI versions of `top` and work in a similar way, but via a point-and-click interface. One other tool deserves special mention, though: `xkill`. This program is a part of the X server, and it's used to terminate a single X window. In a GUI command prompt window, type `xkill`. A message appears in your shell:

```
Select the window whose client you wish to kill with button 1....
```

The mouse cursor changes appearance, and when you left-click on a window, that window closes, terminating the process that controls it. Using `xkill` can be particularly handy if you can see a window that's not responding and want to kill it. You needn't use `top`, `ps`, or any other tool to locate the process. Just type `xkill` and click in the target window.

Using the `kill` and `killall` Commands

Although you can use `top` or other utilities to kill processes, the most basic way to accomplish this goal is to use a utility called `kill`. You optionally pass `kill` a signal by name or number by preceding that name or number with a dash (-) or -s. (If you omit the signal name or number, `kill` sends a TERM signal.) You also pass the PID number of the process to which you want to send the signal. For instance, suppose you've identified a process with PID 2872 that's misbehaving. You decide to terminate it with a TERM signal. Any of the following commands will do the job:

```
$ kill 2872
$ kill -15 2872
$ kill -s 15 2872
$ kill -TERM 2872
$ kill -s TERM 2872
```

If this doesn't work, you can bring out the industrial-strength cleaner and use a KILL signal (signal 9) instead of TERM (signal 15); just make the appropriate substitution in any of the last four commands.

Of course, to use `kill` in this way, you must first identify the PID of the process you want to terminate. You can do this by using `ps` or `top`, as described earlier in this chapter. If you're certain that only one instance of a program is running, or if you want to kill *all* running copies of a program, you can use `killall` instead of `kill`. This program works much like `kill`, but it takes a process

name rather than a PID. For instance, if the program you want to terminate is called `buggy`, you could kill it with a KILL signal like this:

```
$ killall -s 9 buggy
```

WARNING! Some Unix-like OSs have a `killall` program that works differently. Instead of killing all processes of a given name, these `killall` utilities kill all running processes, effectively shutting down the system. If you're using an unfamiliar system, you should verify what `killall` does before using it. Try typing `man killall` to read its man page.

As with terminating processes with `top`, you can use both `kill` and `killall` as an ordinary user to kill the processes you own—that is, those that you've started as that user. You can't kill other users' processes or system processes, though. For instance, an ordinary user can't terminate most servers in this way. To do so, you'll need to run `kill` or `killall` as `root`. The superuser can terminate any user's processes using these utilities.

Adjust Process Priorities

Killing gunky processes is often helpful or even necessary for keeping your system running smoothly; however, it's a rather radical action. Frequently, less extreme measures will work as well, particularly when the problem is a process that's consuming too much CPU time. In such cases, reducing a process's priority can work wonders. Doing so requires knowing a bit about process priorities, so I begin there. You may want to identify a process's current priority, so that topic's up next. Actually adjusting a priority can be done in two ways: starting a program with a non-standard priority or changing the priority of a running process.

What Is a Process Priority?

As already described, the Linux kernel is in charge of doling out CPU time to all processes. When there's not enough CPU time to go around (that is, when the load average climbs above 1.0), Linux must ration CPU time. A process priority is a number between -20 and $+19$ that represents the importance of the process. The default priority if you don't specify otherwise is 0. Confusingly, negative numbers correspond to processes with high priority, whereas positive numbers represent processes with low priority. For instance, if `supercrunch` is running with a priority of -10 and `megaguzzle` has a priority of $+10$, `supercrunch` will get more CPU time than `megaguzzle`, assuming both are requesting 100 percent of the available CPU time.

The kernel doesn't give processes with higher priorities (that is, lower priority numbers) absolute precedence over those with lower priorities. For instance, in the previous example, *megaguzzle* will get *some* CPU time; *supercrunch* won't get it all. The priority numbers do influence how much CPU time each process will get. For example, keeping *supercrunch* constant at a priority of -10, changing *megaguzzle*'s priority from 0 to +10, will reduce its CPU time, and changing its priority again to +15 will reduce its CPU time still further. The extreme values of -20 and +19 represent near monopolization of CPU time and deference to everything else. Chances are you'll never run anything with a priority of -20, because that would interfere too much with other processes. You might use +19, though. A CPU-intensive but low-priority task can benefit from such a setting. For instance, distributed computing programs like *Folding@Home* consume a lot of CPU time, but chances are you don't want such processes to interfere with your normal use of the computer. Running them with a priority of +19 means that they'll take very little CPU time when any other program needs it, but they'll still be able to use all those CPU cycles between keypresses to good effect.

NOTE: Programs can adjust their own process priorities downward (that is, to higher numbers). Some low-priority CPU-intensive programs, including *Folding@Home*, do so automatically when launched.

Identifying a Process's Priority

If a process is consuming a lot of CPU time, you may want to check its current priority to ascertain whether or not it's actually causing problems for other processes. You can accomplish this goal with *ps* or *top*.

To identify a process's priority with *ps*, you must pass an additional option to the command: *l* (that's a lowercase *L*). Typing *ps l* will do the job for processes associated with the current shell, but to see all processes and their priorities, you should type *ps axl*. Because this output will be long, you might want to pipe it through *less* (as in *ps axl | less*) to peruse the contents or use *grep* to search for a specific process (as in *ps ax | grep xfig*) to find the *xfig* process.

However you do it, the *l* option adds several new columns to the *ps* output. In this new output format, the PID is the third column and the priority code is in the sixth column and labeled *NI*, which is short for *nice*, the command that's used to launch a program with a specific priority. (If you use *grep* to search for a specific process, as just suggested, the column headers will be stripped out; thus, you'll need to count the columns rather than rely on column headers.) Most programs you see in the output will have priority values of 0, since that's the default. Some have higher or lower values, though. Typically, those with

negative values (that is, high-priority processes) are important kernel or system processes.

The `top` utility also displays process priorities in a column labeled `NI`. (Check Figure 8-3; it's the fourth column.) Because `top` is designed to show just a few processes, you might not be able to locate a specific process—but if you're hunting for misbehaving processes, the CPU hogs will float to the top of the `top` output, so this can be a good way to identify CPU hogs *and* identify their priorities. Figure 8-3, for instance, shows the top CPU user to be `FahCore_65.exe`, which is consuming 97.4 percent of CPU time but has a priority of 19. In other words, this process is only consuming CPU cycles when they aren't in demand by other processes.

Starting Processes with Non-Standard Priorities

Prevention is usually better than a cure, and in that spirit, Linux provides a simple command to start programs with a desired priority: `nice`. To use this command, type its name, an optional priority adjustment, the name of the program you want to launch, and its arguments, in that order. The priority adjustment takes the form `-n priority` or `-priority`, where *priority* is the priority number from -20 to 19. (This second form is considered obsolete and may not be supported in the future.) If you omit the priority adjustment, the default is to use a value of 10—that is, to reduce the priority about halfway to the minimum. For instance, suppose you want to launch `supercrunch` with a priority of 10 and have it load the `celery.sch` file. Any of these commands will do the trick:

```
$ nice supercrunch celery.sch
$ nice -n 10 supercrunch celery.sch
$ nice -10 supercrunch celery.sch
```

Launching a program with reduced priority in this way can help avert problems, but it's not usually necessary. Most Linux programs are well behaved and won't consume CPU time that they don't need. In fact, if you're directly interacting with a program (such as a word processor, a Web browser, or a mail reader), chances are you want the program to have normal priority, because it will demand CPU time mainly in response to your own commands. If it's given lower priority, a background process might demand CPU time just when you're typing or otherwise interacting with the program, and it will respond sluggishly.

Generally speaking, `nice` is best applied to programs that consume lots of CPU time but that don't need much human interaction, such as tools to create MP3 or OggVorbis files, convert between video file formats, perform CPU-intensive simulations, or donate unused CPU time to your favorite distributed computing project. Such programs tend to take several minutes or even hours to complete their work, during which time you might want to use the computer for something else. By starting such programs using `nice`, you can use other programs at full speed; the `niced` programs will consume little or no CPU time when the programs with which you're interacting need it. Because programs like mail readers, Web browsers, and word processors really consume very little CPU time, using `nice` in this way won't greatly impact the performance of the programs you run with it, but it can greatly benefit other programs' performance.

You can also use `nice` to increase a program's priority (that is, give it a negative value). This will give the program the authority it needs to push other processes around, rather like the proverbial 500-pound gorilla. This can be handy with certain tasks that are timing-critical, such as audio recording programs—if such a program doesn't get its slice of CPU time at the right moment, the resulting audio can be choppy. Because of the potential for abuse of `nice` when used to increase priorities, only `root` may pass negative priority values to the program. If you want to run an ordinary user program with increased priority, you can either combine `nice` with `su` (described earlier, in “Using Local Startup Scripts”) or you can run the program normally and then increase its priority using `renice`, which is described next.

Fortunately, in practice it's seldom necessary to run user programs with boosted priority. Modern computers are fast enough that even most time-critical tasks, such as audio recording, can be done at normal priority. If you encounter stuttering audio, dropped frames in video recording or playback, jerky game play, or similar problems with real-time media, though, you may want to look into this matter. Alternatively, you might consider shutting down or reducing the priority of any other CPU-intensive programs that are running.

Modifying a Running Process's Priority

Preventing process priority problems by careful use of `nice` is great in theory, but in practice, it's sometimes not enough. Perhaps you didn't realize how much CPU time a program would consume until after it was launched, or perhaps it was launched by another user or by an automated script. In any event, you can modify a process's priority by using the `renice` command. In many respects, `renice` works just like `nice`, but it operates on processes that are already running. To use it, you type the command, a priority number, and parameters to identify the processes you want to modify. You can specify processes in one or more of three ways:

- ✓ **By process ID**—Use the `-p` parameter to specify a PID number, as in `-p 2387` to modify the priority of PID 2387. You can omit the `-p` parameter and specify a number alone to have the same effect.
- ✓ **By group ID**—Use the `-g` parameter to affect all processes associated with a specific group ID (GID), as in `-g 107` to change GID 107's processes. You can find mappings of GIDs to group names in `/etc/group`.
- ✓ **By username**—The `-u` parameter enables you to specify usernames, as in `-u sally harry` to affect processes owned by sally and harry.

You can combine multiple options. For instance, if you want to change the priority of all of the processes, groups, and users just described, you might type something like this:

```
# renice 7 -p 2387 -g 107 -u sally harry
```

This command changes all of the affected processes to have a priority of 7. Note that this example uses a hash (`#`) as a command prompt, indicating that it's typed by root. This is because the `-g` and `-u` parameters are usable by root, but their use by ordinary users is quite limited. The reason is that Linux's security model doesn't permit users to modify the priority of each other's processes, root being the one exception to this rule. After all, on a multiuser system, you don't want harry to reduce the priority of sally's processes without her permission! Similarly, users may only decrease their own processes' priority (that is, give them higher numeric values); they can't increase their processes' priority.

In practice and as an ordinary user, you're likely to use `renice` like this:

1. Locate the PIDs of one or more processes that are interfering with other programs by being such CPU hogs. For instance, if you believe that hoggy is such a process, you might type `ps ax | grep hoggy` to locate its PID number. Suppose you find the PID number is 2387.
2. Using the PID number obtained in step 1, call `renice`, passing it a suitable priority code. Typically, you'll be able to use numbers between 1 and 19. For instance, to greatly reduce a process's priority, you might type `renice 10 -p 2387`, which sets the priority to 10.

At this point, the process you've adjusted should begin consuming less CPU time. If you want to increase a process's priority, you can do so, but you'll need to do the job as root. In fact, you'll need to acquire superuser privileges to restore a process's priority to its original value—as an ordinary user, you can only decrease processes' priorities, even if they've already been decreased using `renice`.

Summing Up: Keeping Processes in Check

Considered as a dynamic system, a Linux computer consists of a kernel that manages processes. Even a system with no user programs active is already running several processes, because a bare Linux system includes many processes to handle logins, to manage printers, and so on. These programs are started through SysV startup scripts, and modifying which scripts run can change the character of a system—create a server or a desktop from a generic Linux system, for instance. Aside from login scripts, degunking Linux processes involves locating programs that should not be running or that are running poorly and dealing with them. Programs can run out of control by consuming too much CPU time or memory, and bringing them under control can involve reducing their CPU consumption or killing them. Tools such as `top`, `kill`, `nice`, and `renice` can help in these tasks, so understanding their basics is important to keeping Linux processes degunked.



Account Degunking

Degunking Checklist:

- ✓ Understand how accounts are used in Linux.
- ✓ Know the common types of accounts and common specific accounts.
- ✓ Create and delete accounts to suit the needs of your system, using both text-based and GUI tools.
- ✓ Know why passwords are important for system security and how to create a good one.
- ✓ Set passwords for your accounts using both text-based and GUI tools.

Accounts are central to Linux's operation, and particularly to many aspects of Linux security. When you use Linux, you're using it via an account. Although these accounts are critical to Linux operation, they can become a source of gunk. This can be a result of unnecessary accounts being created, so you should be familiar with the accounts on your system and know when to delete unused accounts and how to do so. (Don't go deleting accounts randomly, though; even many mysterious-sounding accounts are actually required for normal operation.) A second major type of account gunk relates to passwords. If an account has a poor password, or if the password is discovered by outsiders, it can be used to gain entry to your system. Thus, you should know how to select a good password, how to set it, and what steps you can take to prevent your password from falling into the wrong hands.

Why Use Accounts?

Particularly if you're used to Windows 9x/Me, OS/2, Mac OS 9.x or earlier, or certain other operating systems (OSs), you may be wondering what the big deal is about accounts. These OSs work just fine without accounts, which may make you think that accounts are a rather gunky concept. In fact, although simple OSs frequently do without accounts, they serve several important functions in Linux:

- ✓ **Multuser configurations**—Accounts enable you to keep configurations for multiple users on a single system. In the early days of Unix, this was important because computers were big expensive systems with many users at a company, university, or other organization. Although desktop systems today often have but one user, multuser systems are still common. Perhaps your home system is used by everybody in the family, or maybe your office environment can be simplified if employees can work at any computer in the office. Accounts help with this, by keeping users' files and configurations separate.
- ✓ **Multiple simultaneous users**—In addition to simply providing better support for multiple users, accounts can help handle multiple *simultaneous* users. That is, two or more people can use one computer at the same time, via network connections or specialized hardware. Without accounts, changes one user makes to desktop configuration files or the like could easily confuse the other user's display. Accounts help keep things separate, simplifying such uses of a system.
- ✓ **Bookkeeping**—Accounts enable you to track system usage by different users or by different processes. If one user is using too many system resources, you can easily spot this fact and talk to the user about it or cut off system access to prevent further abuses. This feature is most important on

large multi-user systems, as opposed to home systems or even those used in small businesses.

- ✓ **Security levels**—Linux makes a clear distinction between system administration and day-to-day use, and accounts help with this. In particular, the root account is the only one that's permitted to perform certain actions, such as add users, directly access most disk devices, and change software stored in standard locations. This protects a multi-user system from accidental or intentional damage by ordinary users. Additional security levels can be implemented for other accounts or groups of accounts—for instance, certain users can be granted read/write access to a modem, enabling them to use the modem while denying other users that ability.

In fact, the benefits of accounts are great enough that the popular desktop OSs (Windows and Mac OS) have been drifting toward a more Unix-like model for years. Windows XP and Mac OS X try to hide their use of accounts by default, but they lurk just beneath the surface and can be easily uncovered for those who want to take full advantage of them. Most Linux distributions make explicit use of accounts mandatory. If you find this Linux feature strange or annoying, give it a bit of time. Although accounts do place obstacles in your way for handling some tasks, those impediments are there for valid reasons (mostly security related).

WARNING! *It's possible to use most Linux distributions using nothing but the root account, and new Linux users sometimes try to do so. Because of the power of this account, though, you shouldn't use it except when it's absolutely necessary. A simple typo as root can easily wipe out the entire OS! Unfortunately, at least one distribution (Linspire) uses root as the default user account. For this reason, I recommend against using Linspire, or at least reconfiguring it to use a more conventional Linux user account setup.*

GunkBuster's Notebook: Using Network Accounts

This chapter describes accounts that are maintained locally on the computer on which they're used. Such accounts are the norm on home computers and those used in small businesses. Large businesses, universities, and other sites with dozens or more computers, though, frequently make use of network accounts. In this configuration, one computer (and possibly backup systems for it) functions as a *login server*. This computer maintains account databases for the other computers on the network. Rather than define all accounts locally, most computers defer to the login servers for authentication and other account-related tasks.

If your network uses a login server configuration, much of the information in this chapter is still relevant. Typically, system administration accounts, special system accounts, and server accounts are all defined locally. Thus, cleaning them up as described in this chapter makes sense. User accounts, though, may not appear in the local `/etc/passwd` and `/etc/shadow` files. To degunk these accounts, you must do the job on the login server system. These accounts usually aren't stored in `/etc/passwd` and `/etc/shadow` files. Precisely how they're stored and how you would maintain them differs from one protocol and one network login software suite to another. This topic is well beyond the scope of this book. In fact, if you have such a system, chances are you've got a network guru on your payroll whose responsibilities include degunking this system.

Common Accounts That Are and Are Not Needed

When you installed Linux, your distribution created a bunch of accounts. These accounts are defined in two files: `/etc/passwd` and `/etc/shadow`. The first of these files holds the basic account information, and the second one holds passwords and supplementary information. You might want to peruse the `/etc/passwd` file to see what accounts are defined on your system. If you spot accounts that you don't need, you should delete them, as described later, in "Create and Delete Accounts." Generally speaking, accounts fall into four categories: system administration accounts, special system accounts, server accounts, and user accounts.

NOTE: Account details vary from one distribution to another. If you see accounts that aren't described here, or if you don't see accounts that are described here, don't panic! Chances are you're just seeing distribution-specific idiosyncrasies. There is an off chance, though, that an account is a symptom of a break-in, so you might want to investigate that account. Try doing a Web search on the account name and a couple more keywords, such as **Linux** and **account**.

Perusing Password Files

To begin, you should know a bit about the format of the Linux password files, or at least `/etc/passwd`. You don't need to know everything about this file, though—just enough to identify usernames and a few other bits of information. Start by perusing the file by typing `less /etc/passwd` at a shell prompt. The result should be a series of lines like this:

```
root:x:0:0:root:/root:/bin/bash
```

These lines go on for a ways; even a system with just one user account will have 30 or 40 lines, each defining a single account. Each line consists of seven colon-delimited fields:

- ✓ **Username**—The first field (`root` in this example) is the username.
- ✓ **Password**—The second field (`x` in this example) is nominally for the encrypted password; however, as a security precaution, all modern Linux distributions store passwords in `/etc/shadow`, which can't be read by ordinary users. An `x` denotes that the password is stored in this way.
- ✓ **UID**—The third field (a `0` in this example) is the user ID (UID), which is a number associated with the account. Linux uses the UID internally rather than the username for most purposes, but translates back and forth transparently. If you use one of the rare utilities that requires UIDs rather than usernames, you may need to look up this number in `/etc/passwd`.
- ✓ **GID**—The fourth field (another `0` in this example) is a number associated with the primary group of the user. Linux groups can be given group permissions to files and so can be a part of the overall Linux security system.
- ✓ **GECOS**—The fifth field (`root` in this example) is known for historical reasons as the General Electric Comprehensive Operating System (GECOS) field. It often holds a user's real name, or sometimes several comma-separated sub-fields with the user's office number, phone number, and so on. In other words, it's something of a catch all field to hold miscellaneous information.
- ✓ **Home directory**—The sixth field (`/root` in this example) specifies the user's home directory.
- ✓ **Shell**—The seventh and final field (`/bin/bash` in this example) points to the user's default shell—that is, the program that will be run automatically when the user logs in using text-mode tools or when a user launches a command prompt utility in X.

In terms of tracking down gunky accounts, you must pay attention to the account name, the UID, the home directory, and the shell. Frequently, the account name will be enough to tell you that the account is gunky; for instance, you might recognize it as an account you created months ago for testing purposes. Such accounts can be deleted. The UID is important for identifying the account type, as described in the upcoming sections. Also, if a single UID is reused, that can spell trouble, also described shortly. The home directory and shell can also provide clues to an account's function. Certain types of accounts

should have certain values for these fields, and other values could be signs of a break-in.

Linux UID numbers have special meaning that deserves some description. System accounts (that is, what I'm calling in this chapter system administration accounts, special system accounts, and some accounts for servers) have UIDs below 100. Ordinary user accounts have numbers that begin with 100, 500, or 1000, depending on the distribution. For those systems in which user accounts' UIDs begin at 500 or 1000, server accounts' UIDs may fall between 100 and the lowest user account UID. These numbering conventions have no special meaning to the Linux kernel, unlike the UID value of 0 for root, but these conventions can help you identify the intended purpose of an account.

TIP: After installing Linux, back up your `/etc/passwd` file. As root, copy it to a subdirectory of `/root` that only root can read, or copy it to a floppy disk. You can then use this "pristine" account database as a comparison point. If in the future you spot any accounts you don't remember, you can check the original files to see if they were present when the system was new. If they were, you can be fairly sure that the account's presence isn't a sign of a system compromise. If they weren't present originally, they might be a sign of a break-in, or a sign of innocent activity, such as an account added to support a server you've installed.

System Administration Accounts

System administration accounts are those that are used for or otherwise associated with managing the computer rather than day-to-day operations. The most important of these accounts is root, which has a UID of 0. In fact, *any* account with a UID of 0 is effectively the root account, whether or not it goes by that name. For this reason, one check you should perform is to verify that root is the only account with a UID of 0. Crackers sometimes break into a computer and then add an account with a UID of 0, or change the UID of another account to 0, so that they can break in again without actually using the root username or knowing its password. If you think your system has been compromised in this way, disconnect it from the network *immediately* and begin fixing it. The safest way to proceed is to completely reinstall Linux and tighten security. Unfortunately, you might not know how the miscreant got in, so tightening security can be a crapshoot. Tools like `chkrootkit` (<http://www.chkrootkit.org>) might be useful in identifying how your system's security was breached, though.

In addition, most default Linux configurations include several accounts that are intended for administration but that may or may not be used:

- ✓ **adm**—This account is a sort of secondary administrative account. It's present on most Linux systems but is normally disabled. It's seldom used and can usually be safely deleted.
- ✓ **sync**—This account exists to call `/bin/sync` as the login shell. This utility prepares a disk for being shut down. It's not normally called explicitly, and this account is something of a relic of bygone times. It can usually be safely deleted.
- ✓ **shutdown**—This account calls `/bin/shutdown` as its login shell. The effect is that if you log into this account, the computer shuts down. This can be handy if you want to be able to shut down the computer remotely with minimal fuss, but if you don't want to use this functionality, you can safely remove this account.
- ✓ **halt**—This account is like the shutdown account, but it calls `/bin/halt` rather than `/bin/shutdown`. It can usually be safely deleted.
- ✓ **operator**—Like `adm`, this account is a seldom-used supplemental administrative account. It can usually be safely deleted.
- ✓ **postmaster**—This account is required of computers that run mail servers on the Internet at large. If your computer doesn't do this, you can safely delete this account.

In a default configuration, these accounts are all disabled—you can't log into them. This is accomplished by setting an asterisk in the password field in `/etc/shadow` or by setting the login shell to something that won't work as a shell, such as `/bin/false`, which terminates as soon as it's called. In a disabled state, the accounts pose little risk. The problem is that crackers who manage to change the account configuration could enable the account, turning it into an easier way to get back into your system and perhaps do more damage once they are using that account. Thus, removing these accounts from your system is generally a good precaution.

WARNING! Before deleting any sort of administrative account, back up `/etc/passwd` and `/etc/shadow` by copying them to `/root`. That way, if you make a mistake, you can restore these accounts by restoring the original files from the backups. Be sure that the backup is well protected, though. Ideally, it should be in a subdirectory that can only be read by `root`.

Special System Accounts

Special system accounts are those that exist to serve some special purpose for the computer. Typically, Linux runs a program using the account as a way of giving that program reduced privileges, compared to running it as `root`. These

accounts might also be used simply for bookkeeping purposes—to give certain files unique ownership, for instance. The following are some notable special system accounts:

- ✓ **bin**—In theory, this account is associated with binary files (that is, executable programs). In practice, this account is seldom used on modern systems and is a good candidate for removal.
- ✓ **daemon**—This account is used by some daemons—that is, programs that run in the background, such as servers. It might be safe to remove, but if you install a daemon that needs it, its absence could cause problems.
- ✓ **lp**—This account owns files printed using the old-style Berkeley Standard Distribution Line Printer Daemon (BSD LPD) printing system. If your distribution uses the newer Common Unix Printing System (CUPS), you can safely delete it.
- ✓ **mail**—This account is associated with the Linux mail system. Many Linux distributions rely on a mail server being run locally, even if it doesn't accept external connections. Thus, removing this account is inadvisable, although it might be okay to do so.
- ✓ **news**—This account is associated with local Usenet news spools. Most Linux systems don't run a news server, so deleting this account should cause no problems.
- ✓ **uucp**—The Unix-to-Unix Control Protocol (UUCP) is an old telephone-line networking tool, and this account exists in support of UUCP utilities. Chances are you're not using such tools, so deleting this account should be safe.
- ✓ **man**—In theory, this account is associated with the Linux man page system. In practice, it seems to be little used and so can be safely deleted.
- ✓ **cron**—This account is associated with Linux's cron tool for running applications at regularly scheduled intervals. Although the account doesn't seem to be used much, there seem to be some lingering references to it in a few important tools, so I wouldn't recommend deleting it.
- ✓ **at**—The at utility is closely associated with cron; it runs programs at one future time rather than at regular intervals as cron does. As with the cron account, I can't recommend deleting this one.
- ✓ **games**—This account is used to own files associated with certain games. Not all games use this account, but if you play games on your system, you should probably keep it around.
- ✓ **nobody**—This account is unusual because it usually has a UID of 65534, although it's a system account. It's commonly used when a program wants to run as a low-privilege user. Deleting it is likely to break many programs.

As with system administration accounts, you can remove unnecessary special system accounts; however, you should back up your `/etc/passwd` and `/etc/shadow` files before you do so, in case an account turns out to be more necessary than you first thought.

Accounts for Servers

Many servers require special accounts for themselves. In truth, the line between special system accounts and server accounts can be a fuzzy one; for instance, the `lp` and `news` accounts described earlier are tied intimately to servers. I described them as system accounts because these account names are not identical to the names of the server packages or protocols that use them. Most server accounts are named after their servers or protocols. Examples include `ftp`, `sshd`, `gdm`, `xfs`, `named`, `apache`, `cyrus`, `postfix`, and `ntp`. Their purpose is usually to provide a unique account so that the server can be run under this identity. This practice ensures that the server can't interfere with regular users or with other servers or system tools in the event of a bug or misconfiguration.

Most server accounts have home directories set to `/dev/null` or to a directory associated with the server (such as a mail spool directory for a mail server). Because these accounts aren't intended as user login accounts, their default shells are set to `/dev/null`, `/bin/false`, or some other value that won't function as a practical shell. If you spot a server account that specifies a conventional shell (such as `/bin/bash` or `/bin/tcsh`) as the default shell, investigate further. Such a configuration could be a sloppy account definition on the part of the package maintainer, or it could be a hint that your system has been compromised.

Most server accounts are created by the RPM Package Manager (RPM) or Debian package system when you install the relevant server package, although a few common ones may exist as part of the base Linux installation. One of the problems with these accounts is that they often hang around after their packages are deleted. The result is something like an albatross around the neck—dead weight with all the risks of an unused account.

On the bright side, these accounts are usually configured as non-login accounts, just like special system accounts. This means that they can't be used for illicit logins unless the accounts are first modified. Nonetheless, removing them reduces the risk that somebody will be able to modify them, and thus improves security by a small amount. For this reason, you should delete such accounts if their associated servers are not installed. As with similar changes, I recommend backing up your `/etc/passwd` and `/etc/shadow` files before you perform such an operation.

If you're in doubt about such an account, look for the package that created it. (Chapter 6, "Cleaning Out Unused Packages," describes package maintenance, including package browsing tools.) If a package with the same name as the account exists, you shouldn't delete the account unless you're sure the package is unnecessary and you delete it. If you can't find a package named after the account, then it's not a server account at all (perhaps it's a system account), the server has named its accounts in a less obvious way, or it's a server account for an uninstalled server. If you're certain that the account name is the same as that of a server you're not using, go ahead and delete the account. If not, a Web search may turn up some clues.

A variant on server accounts is program accounts. Although the practice isn't common, a few non-server programs like to have accounts for themselves. Their purpose and the procedures for verifying and deleting them are the same as for server accounts.

User Accounts

The most visible class of accounts to normal users is user accounts. As you might guess, these accounts are intended for day-to-day use by ordinary users. You add most user accounts yourself, so you should have a good idea of what these accounts are. On a typical home or small office desktop system, chances are you'll have just a few user accounts—perhaps just one that you defined yourself.

Some distributions automatically add one or two user accounts themselves. The most common of these is `guest`, which is a typical name for a low-privilege account used to grant unknown users limited access to the system. For instance, you might provide a `guest` account to enable users to log in and run a few programs on a public system at a library or university computing center.

Generally speaking, if you're not actively using a `guest` account, you should ensure that your system doesn't have one. Like other unnecessary accounts, a `guest` account is a small security risk, so if you're not using it, there's no reason to live with that risk. The same is true for any unused user account.

TIP: *Temporarily creating a user account can be a great way to test account features or new software, even on a single-user system. Doing so is not a problem, but if you create such an account, be sure to delete it when you're finished using it, lest it become gunk.*

Create and Delete Accounts

Now that you've got some idea of what accounts your system has, you may want to start managing them. For degunking purposes, this means deleting accounts, but you may also want to create accounts. For instance, rather than share a single account among four people living in one home, you can create four separate accounts, enabling each person to have his or her own configuration and a separate place for files.

Most major Linux distributions ship with their own GUI tools for adding and deleting accounts, and these tools are perfectly adequate for handling typical account management tasks. On the other hand, the traditional Linux tools for account maintenance are text based: `useradd` for adding users and `userdel` for removing users. These tools offer features and flexibility that go beyond the GUI tools, and they can also be used from text-only logins and across all distributions, so knowing something about them is desirable. When you delete an account (particularly a user account), one extra step you should take is to track down that user's files and delete or archive them. The account management tools don't do this automatically, so you'll need to use other Linux programs to do so.

Managing Accounts with GUI Tools

GUI account management tools are attractive to new Linux users because they help guide you through the process of managing accounts. In most cases, though, GUI tools are less flexible than their text-mode counterparts. They also tend to vary a lot from one distribution to another. Most GUI tools are distribution-specific, which means I can't describe every distribution's GUI account management tools in this book. In fact, some distributions, such as Debian, Gentoo, and Slackware, eschew GUI account management tools; although you can install various third-party GUI tools, such as Webmin (<http://www.webmin.org>), on these distributions, the usual method of administering them is with `useradd` and `userdel`, as described in the next two sections.

Instead of trying to cover every possible GUI account management tool, I describe just one: the User Manager (aka `system-config-users`) tool of Fedora and Red Hat. Other tools, such as the Edit and Create Users tool of SuSE's YaST and YaST2, work in a conceptually similar way, but many details differ. In practice, you shouldn't have much trouble using another GUI tool once you're familiar with the basic principles. You will need to know what your distribution's tool is called, though. You might find it on a desktop menu, but many such

menus emphasize user tools rather than system administration tools. If you can't find a GUI tool, read ahead to use the text-mode tools.

To get started with User Manager, you must first launch the tool. From a default GNOME or KDE menu, you can launch the program by selecting Applications > System Settings > Users and Groups from the main menu. The result is a prompt for the root password (unless you've launched another administrative tool recently), followed by the User Manager screen shown in Figure 9-1. If you can't find the appropriate entry on your menu or if you're not running GNOME or KDE, try typing **system-config-users** as root in a shell prompt window to launch the program.

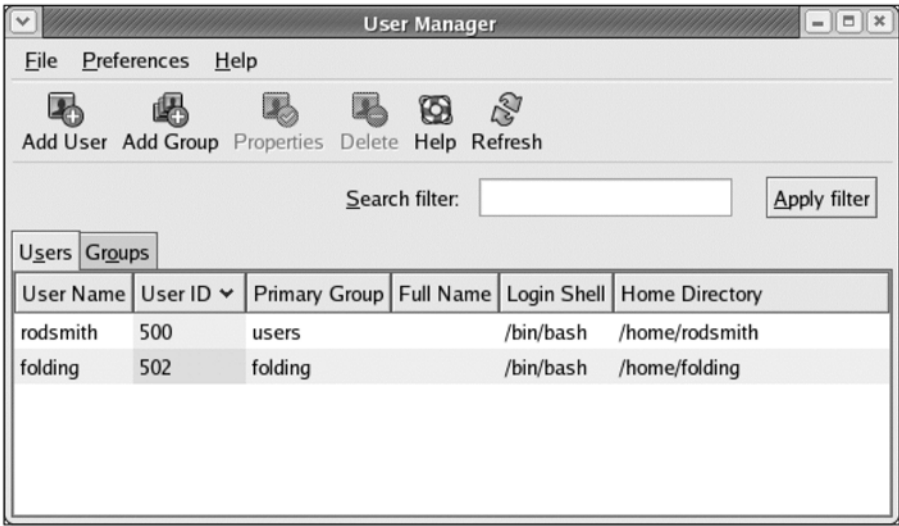


Figure 9-1

GUI user administration tools provide point-and-click access to the local account database.

NOTE: The User Manager program is installed as part of the `system-config-users` package. If you're running Fedora or Red Hat and you can't seem to get the program to run, check for the presence of this package, and if it's not installed, install it. Chapter 6 describes package management, so consult it for more details on how to do this.

Actually administering accounts with User Manager is fairly straightforward. For some operations, you select an account. (There's no need to do this to add a new user, though.) You can then perform various actions, as specified by the buttons in the button bar just below the menu bar:

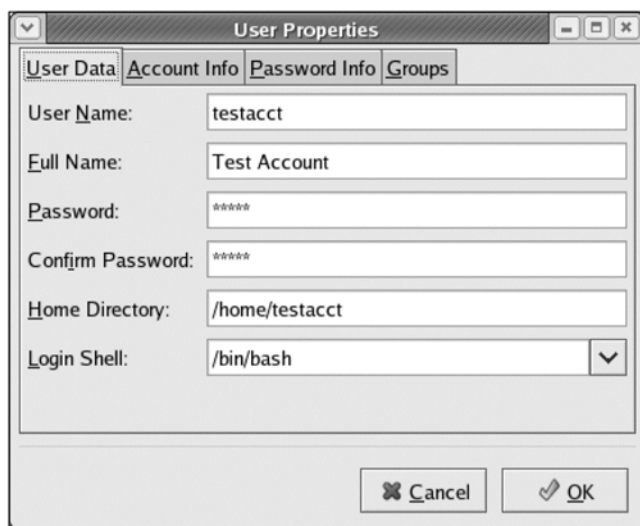
- ✓ **Add User**—Click this button to add a user. The result is the Create New User dialog box shown in Figure 9-2. Enter information in all of the fields. (The Full Name field records information that's stored in the GECOS field of `/etc/passwd`.) In most cases, you should leave the settings in the bottom half of the dialog box alone, as well as the Login Shell field; the default values work fine for most new accounts. Note that when you type a password, it appears as asterisks. This is a normal security precaution, so that nobody can read the password over your shoulder. Check the upcoming section, “Selecting a Good Password,” for information on how to construct a password that's hard to guess or crack from a password file.
- ✓ **Add Group**—You can create a group, rather than a user, by clicking Add Group. Linux manages groups much as it manages users, but chances are you'll only want to add groups on systems with dozens of users, so I don't describe them in detail in this book.
- ✓ **Properties**—You can change key features of an existing account by clicking this button. The result is the Properties dialog box shown in Figure 9-3. The User Data tab enables you to change most of the features you could enter when creating the account. The Account Info and Password

The screenshot shows a 'Create New User' dialog box. It has a title bar with the text 'Create New User' and standard window controls (minimize, maximize, close). The dialog contains the following fields and controls:

- User Name:** A text input field.
- Full Name:** A text input field.
- Password:** A text input field.
- Confirm Password:** A text input field.
- Login Shell:** A dropdown menu with '/bin/bash' selected.
- Checkboxes:**
 - ☒ Create home directory
 - ☒ Create a private group for the user
 - ☐ Specify user ID manually
- Home Directory:** A text input field containing '/home/'.
- UID:** A text input field containing '500'.
- Buttons:** 'Cancel' and 'OK' buttons at the bottom.

Figure 9-2

The Create New User dialog box provides fields in which you enter the most important account information.

**Figure 9-3**

You can change many features associated with an account using the User Properties dialog box.

Info tabs let you change account and password expiration data, which can be handy if an account should expire after a specific period of time (say, if it's for a short-term employee). The Groups tab lets you add the user to an arbitrary number of pre-existing groups. If a program's documentation says that a user should be a member of a group, you can use this tab to add the user to that group.

- ✓ **Delete**—Click the Delete button to delete the account. A confirmation dialog box appears when you do so. This dialog box includes a check box asking if you want to delete the home directory, mail spool, and temporary files. Check this box if you want to delete these files; if not, uncheck the box. The upcoming section “Deleting a User’s Files” describes how to do this job manually. If you want to archive a user’s files, be sure to back them up *before* deleting the account, or delete the account without deleting the files and then back them up and delete the originals manually.

You can access these functions from the File menu, as well as from the button bar, if you prefer.

Using `useradd`

Old-time Linux system administrators generally use text-mode tools for adding or deleting users. Although GUI tools such as User Manager are convenient if you're running a GUI, they don't work from a text-mode login, and they often don't provide access to the more advanced account features.

The first of the text-mode user management tools is `useradd`, which—you guessed it—adds users to the system. In its most basic form, you can type (as root) `useradd` followed by a username to create an account with that username:

```
# useradd jjones
```

This command adds an account for `jjones`. The program returns no output, so the only clues you'll see to its success are that the `/etc/passwd` and `/etc/shadow` files have changed and that a `/home/jjones` directory has been created. Note also that you weren't asked for a password. You must enter one separately, using `passwd`, as described later in this chapter, in "Set Passwords."

As just described, one of the advantages of `useradd` over GUI tools is that `useradd` provides more options. You're not likely to need to use some of the more obscure options, but the ones you might want to use include:

- ✓ **Change the home directory**—You can specify a home directory other than the default one in `/home` by passing the new directory name with the `-d` option, as in `-d /home2/jones` to use `/home2/jones` as the home directory. This option is most useful if you've expanded your system by adding a hard disk and you want to create new accounts on the new disk.
- ✓ **Default group**—The `-g group` option assigns the user to the specified group rather than the default. (The default group is a new one named after the user in Fedora, Red Hat, and some other distributions; it's `users` in most other distributions.) On a small system with just a handful of users, the default group is unlikely to be very important, but group policies can be important on larger multi-user systems.
- ✓ **Additional groups**—You can add a user to additional groups with the `-G groups` option, where *groups* is one or more group names (separated by commas, if you specify multiple groups).
- ✓ **Create home directory**—The `-m` option tells `useradd` to create a home directory for the user, if one doesn't already exist. This option is the default in Fedora, Red Hat, and some other distributions, but many distributions don't create a user's home directory by default. To force the system to *not* create a home directory if your system is configured to do so by default, pass the `-M` option.
- ✓ **Copy skeleton files**—When creating a home directory (as specified by `-m`), `useradd` copies default configuration files from `/etc/skel`. These *skeleton files*, as they're called, serve as the default user configuration files. You can use `-k` to specify another location for the skeleton files if you like, as in `-k /etc/skel2`. This option is only meaningful if you also use `-m`.

- ✓ **Create user-specific group**—As already noted, Red Hat, Fedora, and some related distributions create a group named after the user for each new user. You can disable this behavior by passing the `-n` option to `useradd`.
- ✓ **Create a system account**—The default behavior is to create a normal user account. Such accounts have UIDs of 100, 500, 1000, or above, depending on the distribution. Passing `-r` causes `useradd` to create a system account, which has a UID below the normal user account range, a password that doesn't expire (if one is set), and no home directory. This option doesn't work on all distributions.
- ✓ **Default shell**—The `-s shell` option identifies the default shell for the account, as in `-s /bin/tcsh` to use `/bin/tcsh` rather than the default of `/bin/bash`.
- ✓ **UID**—Ordinarily, `useradd` creates an account with a UID that's one above the highest UID it finds. You can specify another UID by passing one with the `-u uid` option, as in `-u 527` to use UID 527. This feature is most handy if you want to synchronize UIDs across multiple computers.

Once you create an account with `useradd`, be sure to set the password with `passwd` or some other utility, as described later in this chapter in “Set Passwords”—unless of course the account is a system account that should not have a password.

Using `userdel`

From a degunking perspective, you're more likely to need to remove an account than to create one. The tool for this job is called `userdel`, and its basic use is just as straightforward as the basic use of `useradd`—type the command's name followed by the name of the account you want to delete:

```
# userdel jjones
```

Unlike `useradd`, `userdel` supports few options. In fact, the command has just one: `-r`, which causes `userdel` to delete files in the user's home directory and mail spool. If you want to delete user files, either use this option or search for and delete user files manually, as described next. If you want to archive the user's files, be sure to do so before destroying them.

Deleting a User's Files

One of the reasons for deleting user accounts is to clear out the disk space they consume. You can rely on the file-deletion feature of `userdel` or your GUI account management tool to do this job, but sometimes this isn't enough. In

particular, users sometimes leave files lurking outside of their home directories. This is particularly likely to happen if you’ve configured some directory for users to use for file exchange between themselves. To locate and delete such files, you can use a tool that I described in Chapter 3, “Degunking User Files”: `find`. You should review that chapter for details about `find`. To use it to locate files by owner, though, you pass it the `-user username` option:

```
# find / -user jjones
```

This command searches the root directory (`/`)—that is, the entire system—for files owned by the user `jjones`. I specified a root user’s prompt (`#`) because, although this command can be typed as an ordinary user, such users lack permission to read some directories on most systems, so when an ordinary user types this command, some file not found error messages will appear and some directories won’t be searched. Thus, this command is best typed by root.

This form of the command works only when the account still exists. If you try typing this command after you’ve deleted the account, `find` replies with an `invalid argument` error message. All’s not lost, though. If you know the UID that was associated with the account, you can search on that, using the `-uid` parameter:

```
# find / -uid 527
```

This will work, but of course you must know the UID number. If you’ve already deleted the account, you won’t be able to recover this number from `/etc/passwd`. Perhaps you’ll have a backup copy of this file you can use for this purpose. If not, and if you know where to find at least one file owned by the user, you can perform a long directory listing on the file to discover the UID number:

```
$ ls -l /home/jjones/nebula.tif
```

```
-rw-r--r-- 1 527 users 39786 Jan 27 19:12 /home/jjones/nebula.tif
```

The third field, which normally contains the username, holds the UID number if that number can’t be matched to a username. (Recall that Linux uses UIDs internally and maps to usernames based on the contents of `/etc/passwd`. If that mapping can’t be made, Linux reports the UID number.) In this example, the UID number is 527, so you can use that with `find` to locate all the other files with the same UID number.

Once you've found a user's files, you can delete them as you would any other files, using `rm` or a GUI file manager. You may need to perform this task as `root`, though.

NOTE: Ordinarily, Linux lets anybody who has write permission to a directory delete files in a directory, no matter who owns those files. Thus, you **might** be able to clean up a few stray files as an ordinary user, depending on where they're stored. If the user owned the directory and denied others write access to the directory, you must be `root` to clean out the files in the directory. You must also be `root` to clean out such files if the directory's **sticky bit** was set. This bit is a directory feature that changes the rules for deleting files to prevent anybody but a file's owner from deleting it. The sticky bit is often set on directories intended for file exchange and temporary storage.

Set Passwords

Keeping your system clean of unused accounts is an important degunking task, but another degunking task relates to passwords. Specifically, people often set poor passwords on their systems. You should know how to select a good password and set it with either GUI or text-mode utilities. This practice will go a long way toward keeping your system secure. If you're running a multi-user system, be sure all your users know how to select and set good passwords, too. A final password topic is protecting the password; the best password in the world won't keep your system secure if you're sloppy with the password and give it to a would-be intruder.

WARNING! Protecting user passwords is important because a miscreant can use your system as a stepping-off point for attacking others, even if the intruder lacks `root` access. When the intruder's activities are traced back to you, you might take the fall. Potentially as bad, a cracker might use your compromised account and local security bugs to acquire `root` access on your own system. As important as protecting user passwords is, though, you should be doubly careful about protecting your `root` password. This password should **not** be the same as your user password, and you should protect it even more diligently than you protect your user password. If a miscreant gets ahold of your `root` password, you'll be donning virtual biohazard gear, rather than the usual virtual rubber gloves, to degunk your system.

Selecting a Good Password

No doubt you've seen this a dozen times in movies: The bad guy (or perhaps the good guy—good guys tend to do this sort of thing a lot in films) breaks into an office, wearing the sort of black break-into-an-office costume that's only seen in Hollywood. Sitting down at a computer, the black-clad figure

begins typing, and we see a shot of the computer displaying a password prompt. After a try or two, the dark intruder grins: The password has been guessed! It really *was* the name of the computer owner's pet Komodo dragon! The villain (or hero) then begins downloading files, sabotaging the system, or otherwise wreaking havoc.

This sort of scenario, although common in movies, is less common in real life. Real password-based intrusions are frequently based on *dictionary attacks* on the `/etc/shadow` file. In this type of attack, the intruder must first obtain a copy of the file—perhaps it's been stolen from a successful root compromise, or perhaps it's been pilfered in some other way, such as stolen off of a backup tape that was insufficiently secured. In any event, the intruder then runs a large dictionary through the same encryption algorithms used to encode passwords in `/etc/shadow`. If a word creates a match, the password has been found. Given the speed of modern computers, trying every word in English and a dozen other languages, along with common misspellings, word combinations, and other variants, is a trivial undertaking. Sometimes the cracker will attempt to break in using knowledge of the victim, as in the Hollywood scenario. Even this is likely to be done remotely by a program, though, not by manually typing a password into a system.

Other password-discovery methods exist, too. Some are as simple as watching somebody enter a password. Others involve *social engineering*—tricking people into revealing their passwords. Many of these techniques can be thwarted with a bit of care, as described shortly in “Protecting Passwords.”

In order to protect against both dictionary attacks and more personalized attacks, you should know how to select a good password. The goal is to create a password that's as close to random as possible while still being memorable. Unfortunately, these two features are at odds with one another—people, unlike computers, are absolutely terrible at remembering random strings of letters and numbers. One procedure that can help immensely is this:

1. **Select a base.** The base is either two short unrelated words, such as *bunpen* (from *bun* and *pen*), or an acronym that's meaningful only to you, such as *yiwtttd*, for *yesterday I went to the dentist*. If you use an acronym, *do not* use one that spells out a word! I've selected six-letter bases because a subsequent step expands the length a bit and some systems have a maximum password length of eight characters. Modern Linux systems aren't usually limited in this way, though. In practice, an eight-character password should be considered a *minimum* safe length.

2. **Reverse the order.** If you used two words, reverse the order of one word. In this example, the result would be *nubpen* or *bunneq* (the second is better, since *nub* is a word). You could also reverse the order of the acronym, but this is less likely to be helpful.
3. **Randomize the case.** Change the case of some of the letters in the base. This might yield *BUnNep* or *YiWiTD*, among many other possibilities. This step is helpful for Linux, which uses case-sensitive passwords, but some systems use case-insensitive passwords, rendering this step meaningless.
4. **Insert numbers and punctuation.** Insert at least two letters or other non-alphabetic symbols into the password. The result might be *B3UnNe*p* or *Y!iWiT9D*, to name but two examples. Linux accepts both numbers and punctuation in its passwords, but some systems can't use punctuation. This step both moves the password further from the base and expands the potential search space—a cracker who tries a random search will need to use numbers and punctuation, which greatly expands the search space and reduces the odds of finding your password.

In either case, the result should closely resemble gibberish and yet be easy to remember—or at least, much easier to remember than a truly random string of letters, numbers, and punctuation. The password won't appear in any password-cracking dictionary, nor should it be easily generated by a program that tries deviating from the dictionary. In other words, if done right, passwords generated in this way should be next to impossible to guess.

WARNING! *Don't use the bases, much less the passwords, presented here. Although they might make perfectly good passwords in an alternate universe, in this universe they've been published in a book and so might find their way into crackers' dictionaries. This possibility makes them very risky passwords.*

In addition to following this procedure, you should be aware of some *do not*s of password generation:

- ✓ Do not use your own name or username as a base or password.
- ✓ Do not use the name of any family member, pet, or friend as a base or password.
- ✓ Do not use the name of a favorite character from a book, movie, TV show, or anything else; or the title of such a work; or the name of any place or other thing associated with such a work.
- ✓ If using two words as a base, do not use words that are closely related or associated, such as *bird* and *beak*; cracker dictionaries might also link them together, reducing your edge.

- ✓ Do not assume that a word from a foreign language will be proof against detection; as I noted, crackers' dictionaries frequently combine multiple languages together. (Feel free to use acronyms generated in a foreign language or two unrelated foreign-language words as your base, though.)

Most of these measures are intended to protect against attacks targeted at you by people who've done their research. After all, if you have a pet Komodo dragon, somebody targeting you personally may just add your pet's name to the attack dictionary!

Setting Passwords with GUI Tools

For most users, the simplest way to change a password is to use a GUI password-changing tool. As with GUI account management tools for system administration, password-changing tools vary greatly from one distribution to another. These tools can often be accessed from the default desktop environment's menus. For instance, in Fedora 3 running GNOME, select Applications > Preferences > Password. The system displays dialog boxes in which you enter your current password (as protection against somebody changing your password if you happen to leave the computer for a minute or two) and then enter the new password (you must do so twice to be sure it doesn't contain a typo).

KDE, too, provides a password-change tool, which works on any distribution that runs KDE. Type `kcontrol` in a command prompt window or launch the KDE Control Center from the desktop menus. Select the Security & Privacy area and the Password & User Account area within that. You should then see assorted information about your account. Click the Change Password button. As is usual for such tools, you'll be asked once for your current password and twice for a new one.

Sometimes users forget their passwords. When this happens, the system administrator may need to step in and change them to a known value. GUI account maintenance tools, such as Fedora and Red Hat's User Manager, described earlier in this chapter in "Managing Accounts with GUI Tools," can handle this job. Typically, you use the account-editing tool. For instance, in User Manager, you can select an account and then click Properties. On the resulting User Properties dialog box (Figure 9-3), you can type a new password. Be sure to type it twice, once in the Password field and again in the Confirm Password field. When you click OK, the new password should be confirmed.

NOTE: When you change a password as root, you don't need to know the current password. In practice, root never needs to know a user's password, and in fact **should** never know users' passwords. This point comes up again shortly.

Using passwd

The text-mode equivalent of GUI password-change tools is `passwd`. In common use, a user need only type its name to be prompted for a change:

```
$ passwd
Changing password for jjones
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

As when changing a password with GUI tools, you're prompted to enter the current password as a security precaution and then must enter the new password twice to be sure the first entry doesn't contain a typo.

The superuser can change any user's password by adding the username to the call to `passwd`:

```
# passwd jjones
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

When root uses `passwd`, there's no need to enter the current password, just the new one.

Protecting Passwords

Chances are you wouldn't deliberately leave your car keys sitting on the roof of your car while you're wandering through the aisles of your local supermarket. People frequently do the equivalent with their passwords, though, and they may not even realize they're doing so. In practice, passwords can be difficult to protect. Quite a few methods of stealing them exist, and each method has its countermeasures:

- ✓ **Shoulder surfing**—In public and semi-public environments, would-be intruders can lurk and watch you type your password into the computer. The defense is to be alert to others nearby and to try to shield the keyboard with your body, with books, or with anything else while you're typing. If possible, don't use public terminals or enter your password when others are nearby. If you must do so, change your password frequently—and change it from a private location.
- ✓ **Network sniffing**—Crackers frequently install *sniffers*, which are programs that monitor network activity, in an effort to acquire passwords. Network sniffing is a lot like shoulder surfing in that it's an attempt to intercept data during its normal transmission. The usual protection against network sniffing is to use only encrypted protocols for exchanging passwords. For instance, use the Secure Shell (SSH) rather than Telnet or FTP for remote text-mode access and file transfers. Fortunately, over the past decade, unencrypted remote login protocols have become much less common. Unfortunately, many Web sites still transmit passwords in unencrypted form.
- ✓ **Identical passwords**—To reduce memory load, many people employ the same password over and over. This practice has the downside that an intruder who discovers one of your passwords will then be able to use it on many systems. Given the proliferation of Web sites that require passwords, coming up with unique and good passwords for them all can be impossible. I recommend that you either use a Web browser's "keychain" feature to remember them all (which has the drawback of lots of lost passwords if you change browsers or lose your configuration) or use a limited number of passwords for low-security sites and unique passwords for your computer's login and high-security Web sites.
- ✓ **Password cracking**—As described earlier, if a miscreant can acquire your `/etc/shadow` file, the intruder might be able to crack some user passwords. Thus, you should protect this file very carefully. This includes protecting any backups you make of this file, particularly to removable media (tapes, floppies, and so on).
- ✓ **Phishing**—A practice that's exploded in popularity recently is *phishing*, in which the attacker prepares a fake e-mail or Web site that lures users into entering sensitive information. (Phishing attacks often go after credit card numbers rather than passwords, but some might target passwords.) This practice has its historical roots in fake login screens for public computing terminals; an attacker runs a program that looks like a login screen. The program records the login and makes it look like the user entered a typo, when in fact the password has been sent to the attacker. The usual defense against such attacks is a healthy dose of skepticism. Phishing e-mail and Web sites are easy to spot once you know they exist. If you have any doubt,

contact the alleged source of the message, but do *not* use links provided in an e-mail. Manually type the true Web site address into your browser and locate the contact information. If you're using a public terminal and your login fails, it could be a legitimate typo, but if you have any doubt, change your password once you've logged in.

- ✓ **Social engineering**—Phishing is actually a specific type of a much broader attack method known as *social engineering*, in which a would-be intruder asks people for sensitive information. When such requests are couched in convincing enough lies (and sometimes even without that extra flourish!), people tend to divulge lots of information they shouldn't. One well-known ploy is for the attacker to phone or e-mail a target and claim to be a system administrator. This imposter then says that the password database has been damaged and asks the victim for an account password so that it may be reentered. This type of attack is part of why I said earlier that the system administrator should not know users' passwords. More precisely, users should know that the administrator will never ask for them, lest they fall victim to this ploy.
- ✓ **Dumpster diving**—Crackers sometimes rummage through trash to find hidden gems, such as passwords. This risk is one of the reasons that passwords should *never* be written down. Even if you don't throw a password into the trash, it might be found by somebody. If you do write down a password for any reason, you should burn the paper on which it's been written and ideally change the password as soon as possible.
- ✓ **Insecure files**—Storing a password in a computer file can be as risky as writing it down on paper, because passwords stored in this way can be stolen because of a local security breach, a network worm, or outright theft of the computer itself. Unfortunately, password "keychains" have become a practical necessity for those with lots of accounts, and these features store passwords on disk. The better systems encrypt the password file (which of course requires a password to decrypt), but many don't do this. Ideally, you should avoid using programs that store passwords on disk for you. Of course, this may not be practical if you've got dozens of passwords to remember. Your best bet may be to exercise caution and see to your overall system security to minimize the risk of your password files being stolen.

One other general step you can take is to change your passwords frequently. Precisely how often you should do this is something of a judgment call. For a low-security system, once a year or even less might be adequate; for a very sensitive system, single-use passwords (that are valid only once before they must be changed) can be a necessity. In fact, Linux provides tools to help you enforce periodic password changes. GUI tools may provide such options (such

as on the Password Info tab of the User Properties dialog box in Fedora and Red Hat's User Manager), and `passwd` also supports enabling these features. I haven't emphasized them because there are a lot of caveats, and because enforcing such changes is overkill on most small systems.

NOTE: Although this chapter emphasizes Linux login passwords, the principles apply elsewhere, and in fact many of these methods of password theft apply to Web sites and other network password exchanges. The basic principles of password creation and protection apply to any password on any system.

As an individual user, you can take all of these steps to secure your own passwords. If you're administering a Linux system for other users, you should educate those users about the need for password security. Many of these countermeasures are things that only the individual password users themselves can do.

Summing Up: Keeping Accounts Clean and Secure

Linux accounts are vital for the proper functioning of the OS. Simultaneously, though, accounts can become a source of gunk. Most obviously, unnecessary accounts can clutter the system. These accounts can become potential aids to security breaches, and particularly for user accounts, they may chew up disk space. Thus, you should review your accounts and delete those that aren't necessary. Another account-related issue is passwords. Bad passwords are gunk because they can be easily broken. Sloppy password handling can also lead to the password, and thus the account, being compromised. Knowing how to create a good password and how to protect it once it's created can help you keep your system secure.



Network Degunking

Degunking Checklist:

- ✓ Recognize and know how to block Web-based gunk.
- ✓ Understand the problems with blocking Web-based gunk.
- ✓ Know how to block spam, worms, and other e-mail gunk.
- ✓ Remove unnecessary servers and block unused ports to protect against network intruders.
- ✓ Use encryption to protect your data from being sniffed on the network.

The Internet has long been a major source of gunk for all operating systems (OSs). On a worldwide network, some people are bound to be ignorant, selfish, or malicious enough to cause problems for everybody else. Fortunately, you can take steps to head off the worst of these problems. In this chapter, I'll describe three major types of network gunk and the defenses you have against it: Web-based gunk, e-mail gunk, and network security threats. To be sure, there's some overlap in these categories. Most notably, some types of Web and e-mail gunk are security threats. Although this chapter is by no means the final word on network security, I do provide enough information to help you keep the most common types of network gunk from causing you serious problems.

Degunk the Web

To many people, the World Wide Web (aka the WWW or the Web) is synonymous with the Internet. In truth, though, the Web is just part of the Internet; other protocols, such as those used for e-mail, remote logins, setting clocks, and so on, are also part of the Internet. Nonetheless, the Web is an important part of the Internet, and this prominence has led to several types of gunk that dirty the Web. Understanding these types of gunk will help you keep it from causing you too much trouble. Although you can do a lot setting your Web browser's options, I emphasize another tool: a Web *proxy server*, which is something like a filter to remove Web gunk before it even reaches your Web browser. Unfortunately, no matter what method you use to degunk the Web, some gunk is bound to get through, because Web gunk can be hard to differentiate from certain types of legitimate Web content.

Gunk on the Web

The Web can be a remarkably gunky place. At its best, the Web works smoothly and can be a valuable source of information. At its worst, though, the Web can be a slimy, greasy, unpleasant realm—the very definition of gunk. In fact, Web-based gunk comes in many shapes and sizes, including:

- ✓ **Poor HTML**—Most documents on the Web are written in the Hypertext Markup Language (HTML). In theory, HTML is a cross-platform standard that enables any Web browser to read any Web page. In practice, some pages conform better to the HTML standard than do others, and some browsers deal with peculiar HTML better than do others. In practice, your best defense against sloppy HTML is to have several browsers on hand; if one chokes on a page, another might work properly.

- ✓ **Buggy or malicious JavaScript or Java**—JavaScript and Java are two popular scripting languages that are associated with the Web. The idea is that a Web site can include JavaScript or Java code that's run on the user's computer in order to provide extra functionality or offload some of the CPU requirements onto the user's system from the Web server. Unfortunately, JavaScript and Java code can be buggy or, worse, contain malicious content. You can disable your Web browser's ability to run JavaScript and Java, but this will cut you off from some of the Web's legitimate content as well as some of its gunk.
- ✓ **Cookies**—Some Web pages set *cookies* on your computer. These are codes that help the Web server tie your system to its own stored preferences for you or to other data. Cookies can help you save preferences, shop online, and perform other tasks, but cookies can also be used to spy on you. They enable online advertisers to track the commercial sites you visit, for instance. You can have your Web browser block some or all cookies, or you can block cookies using a Web proxy. This may degrade your ability to use some features that rely on cookies, though.
- ✓ **Ads**—About a decade ago, the Internet was considered an advertising-free zone—a sort of refuge for those weary of the constant barrage of advertising. No more. Many Web sites today feature advertising. Some ads are fairly inoffensive, but some use annoying colors or animation. Perhaps the worst of all are the so-called *pop-up* and *pop-under* ads, which display in their own windows that you must close. Modern browsers have options that help a bit by blocking pop-up and pop-under windows that you didn't request, and proxy servers can help filter out more ads.
- ✓ **Inappropriate content**—If you're a parent, you may be concerned about the presence of content you'd prefer your children not see on the Web. Frequently, this is pornography, but violent content or other types of sites might also concern you. Blocking such content is the job of a certain class of Web proxies, frequently known by the name of a specific commercial product for Windows, Net Nanny. Although the original Net Nanny doesn't run on Linux, some Linux proxy servers can fill a similar role.
- ✓ **Bandwidth thieves**—One problem that's common to many of these types of gunk, and to some other content, is that they can chew up your network bandwidth. For instance, if a site has dozens of ads in the form of images, they can take a long time to download, slowing down your access to the information you want. Even if you've got a fast broadband connection, you might notice the slowdown. If you pay for your network connection by the megabyte, Web gunk can actually cost you money! Disabling some features (such as the automatic display of graphics) in your Web browser can help a lot. Web proxies can also help by blocking particular types of content.

Unfortunately, not all types of gunk can be cleaned up. Bad HTML and buggy Java or JavaScript, for instance, are hard to fix. (At least one Linux proxy server, FilterProxy, takes a stab at cleaning up some types of bad HTML, though.) Where your degunking efforts are likely to pay off is in removing buggy or otherwise unwanted content. Specifically, modern Web browsers provide features to disable Java, JavaScript, cookies, and some types of ads. (Chapter 5, “User Settings for Common Applications,” describes these options for Mozilla Firefox.) Web proxies can do the same but are usually more flexible. Web proxies can also protect several browsers or computers simultaneously, so investing in configuring one of them can pay off in terms of simplified configuration if you use several browsers or computers.

Using a Degunking Web Proxy

Web proxies can be great degunking tools. They can degunk the Web for many computers and Web browsers simultaneously, which can greatly simplify your configuration effort if you want to implement identical degunking rules for many computers or browsers. Proxy servers can also do things that are difficult or impossible to do with the options provided by typical Web browsers, such as clean up the Web for childrens’ eyes or block most ads (not just pop-up or pop-under ads).

To use a degunking Web proxy, you must begin by selecting one. Many exist, each with its own strengths and weaknesses. I can’t describe all Web proxy servers, so after a quick review of some popular choices, I focus on configuring and using just one: Privoxy (<http://www.privoxy.org>).

Selecting a Proxy

The first step in setting up a proxy server is selecting one. Quite a few are available for Linux, including:

- ✓ **Squid**—This proxy server, headquartered at <http://www.squid-cache.org>, is designed primarily as a Web accelerator. By caching popular pages and then returning the results from its own cache rather than repeat a lookup, Squid can improve Web browsing speed. This advantage is most noticeable when Squid serves many users, though. In addition to this primary function, Squid provides a great deal of configurability, which makes it the foundation upon which several other proxy servers with other goals are built.
- ✓ **SquidGuard**—This proxy server is built atop Squid and is designed to provide extensive access controls based on the user, the date, the time, the page being accessed, and so on. You can learn more at <http://www.squidguard.org>.

- ✓ **DansGuardian**—This is another Squid-derived proxy filter. It's geared toward blocking sites with offensive content, such as porn or hate speech. The default settings are claimed to be suitable for use by a primary school, but you can change the defaults to be looser or more stringent. DansGuardian filters based mainly on the *content* of the pages rather than a list of specific URLs that are to be blocked. The main DansGuardian page is <http://www.dansguardian.org>.
- ✓ **FilterProxy**—This tool is an engine for filtering Web content in a generic way. It's intended mainly to strip out ads and rewrite poor HTML, but like Squid, it's flexible enough to take on other duties if you're willing to invest the time in configuring it. Check <http://filterproxy.sourceforge.net> for more information.
- ✓ **Privoxy**—This proxy server, headquartered at <http://www.privoxy.org>, is based on an older one called Internet Junkbuster. It's intended mainly as a way of filtering ads (including pop-up ads) and cookies from your Web experience.

Because it's intended for removing the types of Web gunk that most people find most annoying or objectionable (namely, ads and cookies), I describe Privoxy in more detail in the next couple of sections. You might prefer one of the other proxies, though. DansGuardian is particularly worthy of investigation if you want a content filter for children. Although configuration details differ from those of Privoxy, using any proxy server (that is, configuring a Web browser to use it) is done in much the same way for any proxy server.

Running Privoxy

Privoxy's configuration file is usually `/etc/privoxy/config`, but this file relies on several others in `/etc/privoxy` to set most of the details—the main file just sets the broad options, such as the port number that Privoxy uses and all the files it uses to store the detailed configuration. For the most part, the easiest way to configure Privoxy is to use Privoxy itself for the job. You might want to attend to one task before starting Privoxy for the first time, though: Set the port it uses. Like all servers, a proxy server listens for connection requests on a numbered port. This port is set using the `listen-address` line:

```
listen-address 127.0.0.1:8118
```

This default line tells Privoxy to listen using the 127.0.0.1 IP address and port 8118. This configuration is a good one if you want to run Privoxy for a single system; the 127.0.0.1 address is a loopback address, which is used only for local networking. Thus, although Privoxy is a server, the default configuration makes it available only to the computer on which it's running.

Sometimes, though, you want Privoxy to work for many computers. In this case, you should bind Privoxy to the computer's main IP address. For instance, if the computer has the IP address 192.168.1.7, you might change `listen-address` to look like this:

```
listen-address 192.168.1.7:8118
```

This will make Privoxy available to other computers. If the system on which it's running is connected directly to the Internet, this can be a security risk, so be sure you want to make this change. (If your network is protected by a router with firewall features, the risk is lower, but you may want to check your router's configuration to be sure it blocks the port you use for Privoxy.)

In all cases, Privoxy's port number can also be changed. The default of 8118 is fine in most cases. Some clients use port 8000 or 8080 as a default for proxy servers, but this value can be changed in the client, so there's little advantage to changing it in Privoxy. You must remember the port number you used, though.

Once you've changed the address and port number, you should start or restart Privoxy. This is usually done through SysV startup scripts:

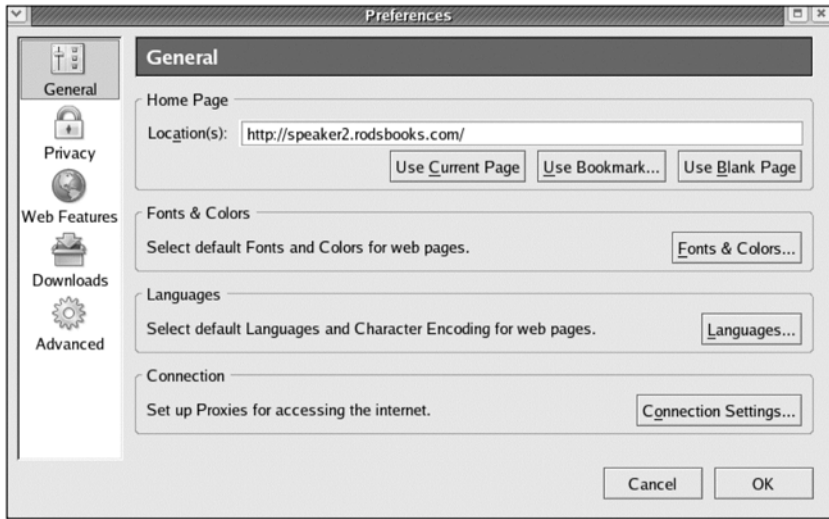
```
# /etc/init.d/privoxy start
```

On some distributions, you may need to change `init.d` to `rc.d`. If Privoxy is already running and you've changed the configuration file, change `start` to `restart`.

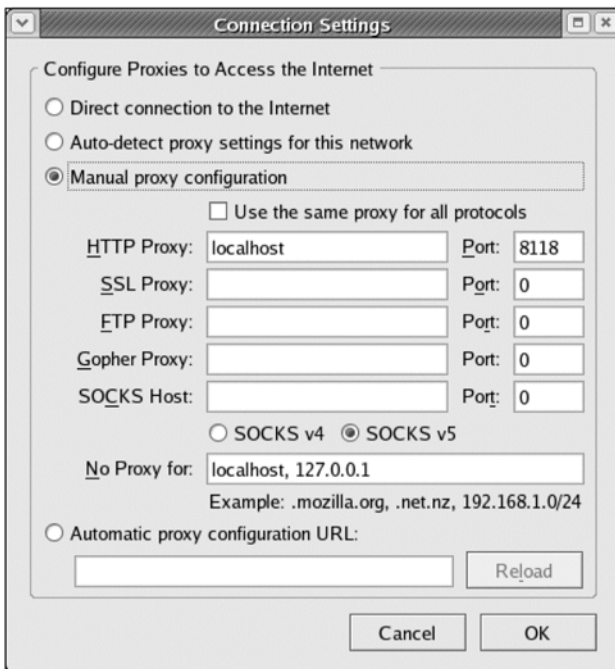
Using Privoxy

To use Privoxy, you must change your Web browser's configuration. Locate a menu option for setting preferences or defaults. For instance, in Mozilla Firefox, select `Edit > Preferences` from the menu bar. The result is a Preferences dialog box, as shown in Figure 10-1. Click the Connection Settings button to get the Connection Settings dialog box, in which you can specify proxies, as shown in Figure 10-2. If you're not using Mozilla Firefox, you may need to hunt a bit to find the equivalent options. For instance, in Konqueror, you would select `Settings > Configure Konqueror` to get the Configure dialog box and then click the Proxy item in the list on the left side of this dialog box.

Once you've found your browser's proxy server settings, you must tell the browser how to use the proxy server. Typically, this is done by clicking an option, such as Manual Proxy Configuration in Figure 10-2, to specify the

**Figure 10-1**

Web browsers' preferences tools provide ways to use a proxy server.

**Figure 10-2**

Web browsers' proxy server options enable you to specify the proxy server and tell the Web browsers when to use the proxy server.

proxy server manually. You then enter the hostname or IP address of the proxy server in the HTTP Proxy field and enter the port number in the matching Port field. (Some browsers, such as Konqueror, require you to click a button to enter the proxy server's hostname or IP address and port number.) Figure 10-2 shows a configuration that should work with a default Privoxy setup—localhost (or 127.0.0.1) as the hostname and port 8118. If you've configured Privoxy to run on another IP address or port, you should make appropriate changes. Most browsers also enable you to specify sites for which the proxy server should not be used. This can be handy if a site doesn't work correctly through the proxy server.

Once you've made these changes to the configuration and dismissed the preferences dialog boxes, try browsing to a site that you know contains lots of gunky ads or other problems. With any luck, you'll find that the ads have disappeared. In the case of Privoxy, they're usually replaced by simple checkerboard graphics that serve as placeholders so that the site's formatting isn't badly damaged. In the case of some particularly gunky sites, Privoxy displays a page of its own, as shown in Figure 10-3. (This specific case is a false alarm—Privoxy interpreted adv in the URL as being a sign of advertising, but it's

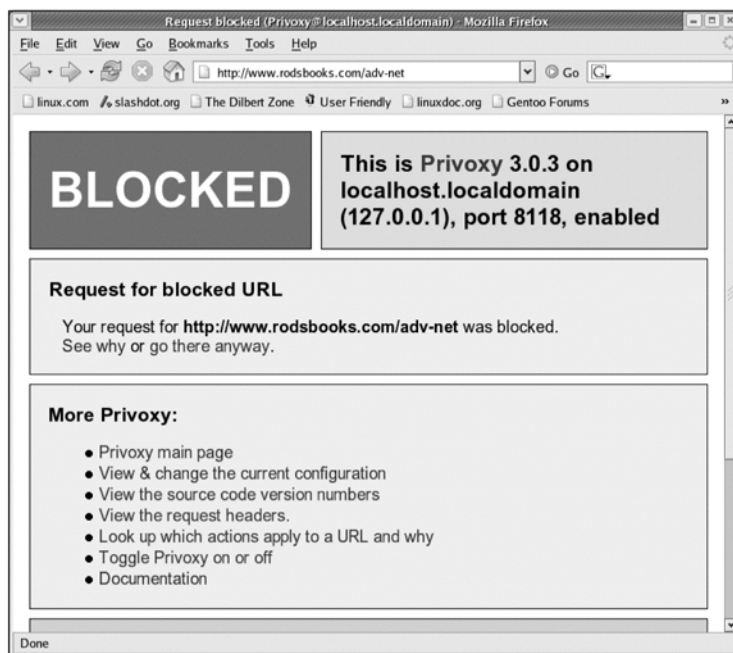


Figure 10-3

When Privoxy blocks an entire page, it displays its own message.

actually short for *advanced*.) If you still want to go to the site, you can click the *go there anyway* link to do so, or you can click *See why* to view the rule that blocked this access—but this rule is expressed in Privoxy’s own configuration language, which may not be easy to interpret. Similar pages result if you mistype a URL or if the site doesn’t respond, but the error message changes to something appropriate.

By removing ads, Privoxy can speed up your Web browsing experience. By removing unnecessary cookies, Privoxy can improve your privacy while browsing the Web. Unfortunately, Privoxy isn’t perfect. If you’re up for it, you can tweak Privoxy’s configuration to try to improve it, but this process isn’t easy. If you want to try, enter **<http://config.privoxy.org/show-status>** in your Privoxy-using Web browser. This brings you to the Privoxy configuration tool. Lots of documentation is available from this tool, but configuring Privoxy at this level isn’t for the faint of heart. In fact, it’s downright tricky, particularly if you’re not already familiar with tools like regular expressions. Thus, I don’t describe tweaking Privoxy’s configuration in this way in any detail.

GunkBuster’s Notebook: Perils of Degunking the Web

One problem with tools like Privoxy is that they can never be 100 percent accurate. The proxy will miss some types of gunk, raise false alarms over non-gunky sites, and generally interfere with the normal operation of certain sites. For the most part, Privoxy and other degunking proxy servers work well, but sometimes they just create their own type of gunk.

The simplest solution to such problems is to disable your browser’s use of the proxy. You can do this on a site-by-site basis by providing an exception list to the Web browser, as noted earlier. Sometimes this doesn’t work well, though, because the problems are with references to a site other than the one you’re viewing. Temporarily disabling the browser’s use of the proxy server can help work around these problems, but of course this opens your browser up to all the gunk until you re-enable the proxy. Another option is to keep a second Web browser configured to not use the proxy server. You can then run this Web browser whenever you run into a site that doesn’t behave well with the proxy server.

Note that the fact that a Web browser’s proxy server can be so easily disabled can be a problem if your intent is to keep teenage children from locating porn sites or the like; they can simply disable their browsers’ proxy server settings to browse directly.



You do have an extra tool in such cases, though: firewall settings, as described in “Blocking Ports from Unauthorized Use.” You could block a child’s computer from browsing directly in a separate firewall system or block outgoing browsing data from anything but the proxy server software. Unfortunately, the simple GUI tools provided by most Linux distributions are incapable of configuring such a subtle set of rules, so you may need to use a more sophisticated GUI tool or design your own `iptables` script to do the job—tasks I don’t describe in this book. If you’re using a separate hardware router, some or all of the configuration is likely to involve it, so you may need to consult its documentation.

Clean Up Your E-Mail

Broadly speaking, e-mail gunk falls into two categories: spam and worms (the latter are also often called viruses). Spam is unsolicited bulk e-mail and is basically an annoyance. Worms are at least potentially more serious because they are malicious programs. (In practice, worms almost always target Windows systems, so you’re fairly safe from them running Linux, but they can still clog your inbox and chew up your network bandwidth.) Tools to block both types of gunk exist, and in fact, the same programs typically handle both types of gunk. Finally, some e-mail is just plain weird—it contains bizarre characters or is formatted oddly. Unless this e-mail also happens to be spam or a worm, its content isn’t gunk; it just *looks* gunky. Knowing how to deal with such e-mail can help you communicate with whomever sent it.

Using Spam and Worm Blockers

Estimates peg the amount of e-mail that’s spam and worms at about 80 percent. This flood of worthless e-mail can easily swamp your legitimate e-mail, making it hard to locate and respond to legitimate correspondents. To help ease the burden, many tools exist to filter spam from your e-mail stream. Some of these tools work with any mail reader, but others are designed to work with just a few mail readers. Once you’ve picked an anti-spam tool, you must configure it to work on your system and perhaps with your e-mail client. As an example, I describe how to get the Evolution mail reader working with SpamAssassin.

Selecting an E-Mail Degunking Tool

You know spam when you see it. Ads for products to enhance or shrink various body parts, solicitations to help somebody smuggle funds from a foreign country, and claims that accounts you never knew you had are about to be closed are all

obvious types of spam—at least, they’re obvious to people who have much experience with spam. E-mail worms are also usually easy to spot once you’ve seen a few. They often claim to be from somebody you know and contain a cryptic message about an attachment you didn’t actually request.

NOTE: For brevity, I frequently refer to “spam filtering” or the like when I really mean “spam or worm filtering.” Most Linux spam filtering tools are equally good at blocking worms, so the two tasks are really just one task from a Linux perspective.

Unfortunately, computers aren’t nearly as good as people at identifying spam. Suffering from a severe lack of human language skills and real-world knowledge, a computer can’t use common sense to rip out the e-mail weeds. Instead, programmers have come up with assorted ways of identifying spam, including creating blacklists of the IP addresses that are most often used to send spam, checking e-mail messages for keywords that are likely to identify spam, and using statistical tests to classify messages as spam or non-spam based on previous classifications. Some of these tools, such as IP address blacklists, work best on mail servers. Others, such as statistical tests, are best employed on e-mail clients. In Linux, some of the most popular spam-fighting tools include:

- ✓ **Mail server options**—All major Linux mail server packages provide spam-fighting tools, such as simple pattern-matching options and checks against IP-based network blacklists. These options are most viable if you’re running your own mail server, so I don’t describe them here.
- ✓ **Procmail**—This program isn’t really a spam-fighting tool *per se*; instead, it’s part of a Linux computer’s local mail delivery system. It can perform pattern matches and so can be used to weed out spam based on keywords or known bad addresses. You can also use Procmail to call various other spam-filtering tools. You’re most likely to use Procmail in your spam fighting if you run a mail server computer (a big topic in and of itself) or if you want to use Linux’s local mail queue capabilities to help integrate mail from several sources (another big topic).
- ✓ **Bogofilter**—Statistical mail filters exploded onto the scene in 2002 and have proven themselves capable spam-fighters. Bogofilter (<http://bogofilter.sourceforge.net>) is a fast and popular example of a statistical spam filter. It can be integrated into a Linux mail queue or called from certain mail readers. Its major drawback is that it requires a sample of spam and non-spam messages to get started, so you should collect your spam for a few days (or longer) before starting to use Bogofilter.
- ✓ **Mailfilter**—Most user-level anti-spam tools work after you’ve downloaded the message and wasted your precious bandwidth. Mailfilter’s claim to fame is that it can delete spam from a Post Office Protocol (POP) account *before*

you download it, thus saving you the connect time. Mailfilter is a simple pattern matcher and so is less effective and harder to maintain than more sophisticated tools like Bogofilter and SpamAssassin, but it can be particularly handy if you've got a dial-up network connection or when a new worm rages out of control, filling your in-box with hundreds of megabytes of garbage each day. Check <http://mailfilter.sourceforge.net> for more information.

- ✓ **SpamAssassin**—This program, headquartered at <http://spamassassin.apache.org>, combines several spam-filtering techniques in a single program, including simple pattern matches, IP-based tests, and statistical tests. For this reason, SpamAssassin is very large and complex, but basic configuration is relatively straightforward. Many Linux mail readers can interface with SpamAssassin for spam-filtering duties.

The spam world is constantly changing, and new spam-fighting techniques are always under development. Furthermore, established programs are updated frequently because spam is a moving target. Once any spam-fighting technique starts to have a significant impact, spammers adapt and find ways to structure their messages to thwart the anti-spam tools. (This is why so many spams use bizarre spellings or punctuation inside words—these measures are attempts to work around pattern matches.) For these reasons, you should be sure to update your anti-spam tools frequently, and periodically review the available tools. Something new may be available and do a better job than an old standby.

Configuring Evolution to Block E-Mail Gunk

Most modern e-mail readers provide the means to filter spam. In Linux, this is usually done by linking the mail reader to an external program, such as Bogofilter or SpamAssassin. Specifically, you create a filter rule that passes all your messages through the external program, which adds a special flag to the message indicating whether or not it's spam or returns a value to the calling program that conveys the same information. The mail reader can then look for this flag and, if it's present, shunt the mail into a separate spam folder or even discard it outright.

WARNING! *Completely discarding suspected spam on an automatic basis is a risky proposition. Because spam filters occasionally produce false alarms, chances are you'll throw away at least some legitimate e-mail if you do this. If you place suspected spam in a separate mail folder, you can periodically review it or simply dive into it if you suspect a message may have been misclassified.*

As an example, I describe configuring Evolution to use SpamAssassin for spam filtering. If you use another mail reader, the procedure will differ in many details. If you prefer to use another spam filter, much of this procedure will be

the same, but the filtering details will differ. In any event, to proceed you should follow these steps:

1. Install both Evolution and SpamAssassin, if they aren't already installed on your system.
2. Launch Evolution and, if necessary, configure it to read your mail.
3. Create a new folder in Evolution to hold suspected spam. (Chapter 5 describes creating folders in Evolution.)
4. Select the Tools > Filters item from the Evolution menus. The result is the Filters dialog box, which summarizes the filters Evolution has installed to sort incoming mail. The default filter set is empty; no filters are defined.
5. In the Filters dialog box, click Add. This produces the Add Rule dialog box shown in Figure 10-4, in which you can create a filter rule.
6. Type a name in the Rule Name field, such as **spam** or **SpamAssassin**.
7. In the If part of the Add Rule dialog box, select the following options:
 - ✓ In the leftmost list selector (which reads Sender by default), select Pipe to Program. (Some versions of Evolution use the words Pipe Message to Shell Command for this option.) When you do this, the other controls in the If area change to reflect options suitable for piping messages to an external program.

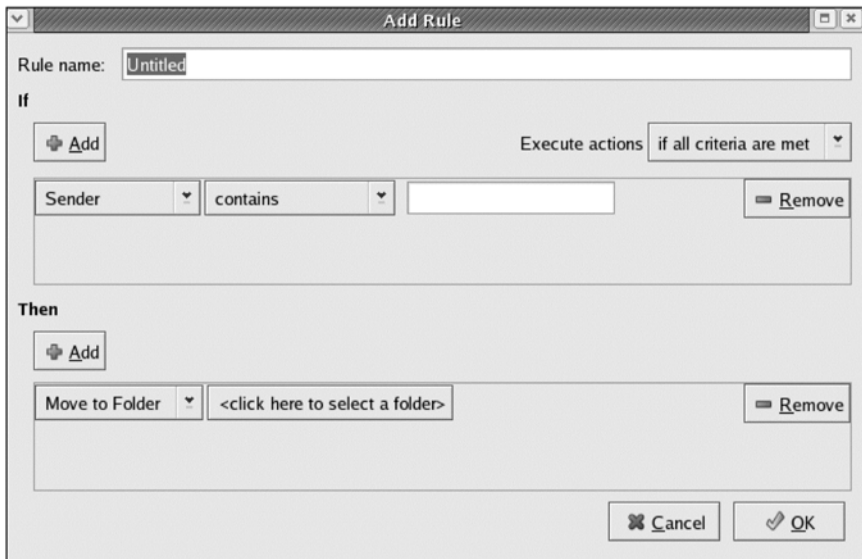


Figure 10-4

Evolution uses filter rules, which can match patterns or call external programs, to automatically redirect e-mail to specific folders.

- ✓ In the text box that appears to the right of Pipe to Program, type **spamassassin -e > /dev/null**. Ordinarily, SpamAssassin performs some tests that require an active network connection. If you read your mail offline, you may want to add **-L** to this command, as in **spamassassin -L -e > /dev/null**, which disables the online tests.
 - ✓ Set the Returns drop-down box to Returns Greater Than.
 - ✓ Be sure the value to the right of the Returns Greater Than button remains set to 0.
8. In the Then part of the Add Rule dialog box, select the following options:
 - ✓ Set the leftmost list selector to Move to Folder. (This is the default value.)
 - ✓ Click the <Click Here to Select a Folder> button and pick the spam folder you created earlier.
 - ✓ Click the Add button just below the Then heading.
 - ✓ In the selector that appears as a result, choose the Stop Processing action.
 9. Click OK in the Add a Rule dialog box.
 10. If you have more than one filter rule, move the spam filter rule to an appropriate point in your list. For instance, you might want to put it after rules to move mail from known good recipients into particular folders, but before rules to move mail from mailing lists into their folders.
 11. Click OK in the Filters dialog box.

At this point, Evolution should use SpamAssassin to filter your e-mail. The key to this procedure is the call to SpamAssassin itself, as created in step 7. You can call another spam filter instead, if you prefer.

Tweaking SpamAssassin's Results

A simple call to SpamAssassin, as just described, uses SpamAssassin's default rules. For best results, though, you should tweak your SpamAssassin configuration. One way to do this is to delve into the SpamAssassin configuration file, which can get a bit hairy if you're uncomfortable with such things. Nonetheless, making such changes can help improve SpamAssassin's performance. You can also pass specific messages to SpamAssassin to have it register their status as spam or non-spam, thus improving the program's ability to correctly identify similar messages in the future.

SpamAssassin uses two configuration files: `/etc/spamassassin/local.cf` is a system-wide configuration file, while `~/.spamassassin/user_prefs` is the user

configuration file. For a desktop system, chances are you'll modify the latter file. In either file, you might want to change a few types of options:

- ✓ **Spam threshold**—SpamAssassin works by assigning scores to messages. The higher the score, the more likely it's spam. (Unlike statistical filters, though, SpamAssassin doesn't attempt to assign true spam probabilities to messages, except as part of its own statistical sub-filter.) You can change the `required_score` value to alter the threshold score for calling a message spam. Lower the score from the default value of 5.0 to make SpamAssassin more likely to call something spam; raise the score to lower the likelihood that the message will be classified as spam. If you find that SpamAssassin is misclassifying too many messages, you might want to adjust this value. Be wary of lowering this value, though; doing so could result in a lot of false alarms.
- ✓ **Change scores**—Instead of changing the overall spam threshold, if a particular rule is causing a message to be incorrectly classified as spam, you can change its score. To do so, use the `score` keyword, as in `score MPART_ALT_DIFF 1.0` to set the score for the `MPART_ALT_DIFF` test to 1.0. A list of rules and their default scores is available from http://spamassassin.apache.org/tests_3_0_x.html, so consult that page. Altering scores in this way requires careful research on the scores and why SpamAssassin is classifying particular messages as spam. (I describe how to do this shortly.)
- ✓ **Whitelists**—A *whitelist* is a list of addresses or other mail characteristics that should give a message an automatic “pass.” For instance, you might want to create a whitelist with the addresses of all your regular correspondents. You can use the `whitelist_from` option to add an address to the SpamAssassin whitelist, as in `whitelist_from emily@example.com` to let through any mail with that return address. If you want to whitelist several addresses, you can either place them all on one line, separated by spaces, or create multiple `whitelist_from` lines. The `whitelist_to` option enables you to create a whitelist based on the recipient address (this might be helpful for mailing lists, for instance), and various other whitelist options are also available. consult the SpamAssassin documentation for details.
- ✓ **Language detection**—The `ok_language` option enables you to specify what languages are acceptable. If SpamAssassin believes the message is in another language, it will be flagged as spam. This option takes one or more two-letter language codes as options, as in `ok_language en fr` to accept English and French messages.

Many additional options are available as well. For more information on these options, type `perldoc Mail::SpamAssassin::Conf`. This produces documentation on the SpamAssassin configuration file that's similar in format to a typical man page.

During its normal operation, SpamAssassin attempts to update itself dynamically. This is particularly important for its statistical rules. When SpamAssassin determines that a message is spam, that message's words are added to the spam word list; when SpamAssassin classifies a message as non-spam, its words are added to the non-spam word list. Unfortunately, this classification is sometimes incorrect. For such cases, you may want to fine-tune the listing. To do so, you should first extract the message to a text file. For instance, in Evolution, select the message and pick File > Save As from the Evolution menu bar. You can then save the message to a text file and perform various actions on it from a command prompt:

- ✓ **Study SpamAssassin's classification.** To study how SpamAssassin is classifying a message, type `spamassassin -t <message.txt`, where *message.txt* is the message's filename. The result is the message being echoed to the screen, followed by a series of lines describing the tests that triggered on the message. You could use this information to tweak the scores on your rules, as just described.
- ✓ **Manually train the statistical filter module.** If a message has been misclassified as spam or non-spam, or if you have a large collection of messages of one type or another, you can pass them to the `sa-learn` program, which takes the `--spam` and `--ham` options to classify messages as spam or non-spam, respectively. For instance, type `sa-learn --ham message.txt` to classify a message as non-spam.
- ✓ **Report spam to online databases.** SpamAssassin's `-r` (or `--report`) option delivers a message to various online spam databases. To do this, type `spamassassin -r < message.txt`. Once this is done, identical messages sent to others who use these tools will be rejected, even if the messages don't trigger other spam-detection rules. I recommend not using this feature until you're comfortable with SpamAssassin, and then only with messages that seem to elude its normal detection system but that are clearly spam.
- ✓ **Add a message to a local whitelist.** You can add a message to a whitelist by using SpamAssassin's `-W` (or `--add-to-whitelist`) option, as in `spamassassin -W < message.txt`. This procedure is similar in effect to using the `whitelist_from` option in the configuration file, but the `-W` option adds more addresses from the headers than you probably would if you did it manually.

Reading Weird-Looking E-Mail

Sometimes e-mail just looks bad, perhaps to the point that it's unreadable. If you've received a message and you just can't make heads or tails of it, look into several possible causes and cures:

- ✓ **HTML displayed as such**—People are increasingly using HTML for their e-mail messages; however, if your mail reader is configured not to parse HTML, you'll see an ugly mess of HTML codes, as shown in Figure 10-5, which shows KMail displaying an HTML message. Some mail readers enable you to view the formatted HTML version by clicking a button or

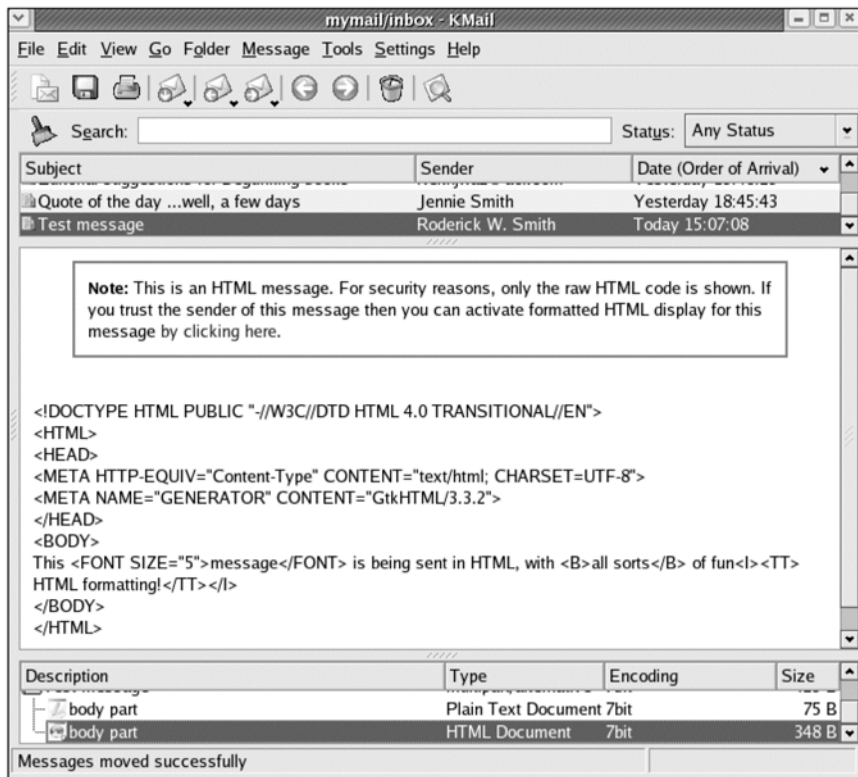


Figure 10-5

HTML e-mail can be difficult to read if your mail reader doesn't parse it by default.

link; however, doing so can open you up to some security risks because such messages may contain *bugs* (links to small graphics files that enable others to track when you read the message) and even malicious Java or JavaScript code. Linux e-mail readers are typically resistant to such tricks, but to be safe, I recommend trying to read through the HTML formatting instead. If a real person sent the message, you might consider asking them not to send HTML-only messages. Note that even messages from ordinarily trustworthy sources may be untrustworthy because Windows worms often send mail to people in victims' address book lists. Although Linux is effectively immune to Windows e-mail worms, playing it safe is better than being sorry.

- ✓ **Ugly or poorly sized fonts**—If your mail reader automatically renders HTML, as Evolution does by default, you might end up with fonts that are ugly, too small, or too large. If the problem is poorly sized fonts, look for a font size option. In Evolution, these options are under the View > Text Size menu. Bizarre fonts, if encoded in the HTML, can be harder to work around. An option to disable HTML parsing, if available, may help. If most or all of your messages are hard to read because of poor fonts, look for a font option in the program's configuration menus; perhaps your defaults are just set poorly.
- ✓ **Foreign fonts**—Most messages with foreign fonts (Cyrillic, Chinese, or others) are spam, so you should try installing spam filters. (At least this is true for people who don't read languages that use such fonts—others should apply common sense!)
- ✓ **Unreadable attachments**—Sometimes people will send attachments that aren't readable because they were created with obscure Windows-only programs or encoded in some odd way. Your best bet is to contact whoever sent you the message and ask for information to help you read it, or to re-send the message in a more conventional file or attachment format. Remember, too, that strange attachments are often Windows worms. The apparent sender might not have actually sent the message; that person's system might be infected with a worm, or another person's system might be infected and be impersonating the apparent sender.

Improve Your Network Security

Throughout this book, I've made repeated references to poor security as a source of gunk. A lot of security issues center on the network, so this is an appropriate place to cover some network security issues. These are removing unnecessary servers, blocking network ports, using encryption, and monitoring your system for signs of intrusion.

WARNING! *The steps described in this chapter are just some important security procedures. Various chapters of this book describe several others, such as keeping your software up-to-date. Entire books have been written on network security, so I can only scratch the surface in these few pages!*

Removing Unnecessary Servers

Dozens, if not hundreds, of server programs are available for Linux. Most of these server programs constitute gunk on any given Linux system. Even if the computer's main function is as a server system, chances are it only needs to run one or two network-accessible servers. The remaining servers that *could* be run

on that system will just take up disk space, consume RAM, and (worst of all) pose a security risk. Every running server program is a potential door into your computer. A bug or misconfiguration could give an intruder a wedge that might be used to change critical configuration files, thus granting the outsider partial or full control of your system. For this reason, you should be careful about running servers unnecessarily and take steps to hunt down and uninstall any that you don't need.

Removing unnecessary servers is really a special case of removing unnecessary software, which was described in Chapter 6, "Cleaning Out Unused Packages." Thus, if you haven't already read that chapter, you should do so. In particular, you should be familiar with your distribution's package management system—typically the RPM Package Manager (RPM) or Debian packages, but perhaps something else. You should use a package browser, as described in Chapter 6, to track down server packages that you don't need and remove them. Beyond this, you can use special tools to track down running servers and remove them.

Tallying Your Running Servers

Network security is important enough that you shouldn't stop with package manager lists. A server might escape your notice in the package descriptions, or it might have been compiled and installed locally and then forgotten. Additional tools can help you locate extra servers. In particular, the Linux `netstat` command is an (almost) all-purpose networking tool that can be used to help track down open ports—that is, the doors that servers use to accept connections from outside.

To use `netstat` to locate open ports, use its `-l` (that's a lowercase *L*) and `-p` options in combination. The `-l` option causes the program to display information on all processes that are listening for connections, and `-p` creates an extra column to identify the program by program ID (PID) number and name. The latter option requires you to be `root` to get all the PIDs and names, so you should run the command as `root`. The output is likely to be quite long, so you should pipe the output through `less` to enable you to page back and forth through the output to better study the result. The final command looks like this:

```
# netstat -lp | less
```

Unfortunately, the output of this command is likely to include lines longer than the usual 80 columns, so you may want to widen your terminal window before you execute it. Figure 10-6 shows the output of this command on one system.

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	*:32769	*:*	LISTEN	2750/rpc.statd
tcp	0	0	*:32770	*:*	LISTEN	-
tcp	0	0	*:netbios-ssn	*:*	LISTEN	3063/smbd
tcp	0	0	*:vnc-760x530	*:*	LISTEN	3009/xinetd
tcp	0	0	*:vnc-800x600	*:*	LISTEN	3009/xinetd
tcp	0	0	*:vnc-950x700	*:*	LISTEN	3009/xinetd
tcp	0	0	*:vnc-1024x768	*:*	LISTEN	3009/xinetd
tcp	0	0	*:sunrpc	*:*	LISTEN	2730/portmap
tcp	0	0	*:x11	*:*	LISTEN	3716/X
tcp	0	0	*:vnc-indirect	*:*	LISTEN	3009/xinetd
tcp	0	0	*:vnc-640x480	*:*	LISTEN	3009/xinetd
tcp	0	0	*:6003	*:*	LISTEN	4058/Xvnc
tcp	0	0	localhost.localdomain:8118	*:*	LISTEN	4225/privoxy
tcp	0	0	localhost.localdomain:ipp	*:*	LISTEN	2937/cupsd
tcp	0	0	localhost.localdomain:5335	*:*	LISTEN	2913/mDNSResponder
tcp	0	0	*:smtp	*:*	LISTEN	3044/sendmail: acce
tcp	0	0	localhost.lo:x11-ssh-offset	*:*	LISTEN	3978/0
tcp	0	0	*:microsoft-ds	*:*	LISTEN	3063/smbd
tcp	0	0	*:x11	*:*	LISTEN	3716/X
tcp	0	0	*:6003	*:*	LISTEN	4058/Xvnc
tcp	0	0	*:ssh	*:*	LISTEN	2967/ssh
tcp	0	0	:::x11-ssh-offset	*:*	LISTEN	3978/0

Figure 10-6

The `netstat` command can produce copious output, but if you know how to trim it down, the output can be a gold mine.

Important columns in the `netstat` command output include the following:

- ✓ **Protocol**—The `Proto` column identifies the protocol being used. This is most commonly `tcp`, `udp`, or `unix` for TCP ports, UDP ports, or Unix domain sockets, respectively. You should be concerned with TCP and UDP ports. Unix domain sockets are entirely local and their use doesn't constitute a security risk—at least, not in the way TCP and UDP ports do.
- ✓ **Local address**—The `Local Address` column identifies the IP address and port on which the server is listening, separated by a colon. In many cases, the IP address is reported as an asterisk (*), meaning that the server is listening on all IP addresses. If the IP address is `localhost`, `localhost.localdomain`, or `127.0.0.1`, it means that the server is listening on the localhost address only, which is a minimal security risk. Ports may be identified by number, or if they're mapped, in the `/etc/services` file, by name.
- ✓ **PID and program name**—The `PID/Program name` column identifies the server by PID number and name. You can use this information to track down the program and, ultimately, uninstall it.

Identifying Running Servers

The trick now is interpreting the information from the `netstat` output. Clues exist in both the port numbers (or names) and in the server names. You might recognize one immediately as the name of a server or protocol. For instance,

near the bottom of Figure 10-6's output, you can find the `sshd` process running on the `ssh` port. You might recognize this as belonging to the Secure Shell (SSH) server and protocol. If a server or port doesn't strike you as familiar, you can use several resources to track them down:

- ✓ **Your local package system**—You may be able to locate the package to which a server belongs. Type `whereis servername`, where *servername* is the name of the server as identified in the `Program` name column. This should identify the location of the server's program file. You can then use your RPM or Debian package tools to locate the server package and read its description to discover what it does.
- ✓ **Your man pages**—Type `man servername` or `man portname` to locate information on the server or port that might exist in the local Linux man pages. This method might not work, but you could luck out and get what you need.
- ✓ **An Internet search**—Use your favorite Internet search engine to perform a search on the program or protocol name. This might or might not produce a result, and it's unlikely to find anything if you enter a protocol number rather than a name.

Once you've identified the server, you can evaluate whether or not you need it. Some, such as an X server for a desktop system, are likely to be necessary. Other common servers you probably don't want to shut down include the port mapper (aka the Sun RPC handler—that is, the `portmap` server and the associated `rpc.statd`) and a printer queue package (most commonly CUPS, but sometimes LPRng or even the BSD LPD program). The `inetd`, `xinetd`, and SMTP servers are all ones you shouldn't shut down lightly, although you might ultimately decide to do so. All other servers deserve careful scrutiny. You might ultimately decide to leave them running, but you shouldn't accept their presence without further consideration or research.

Blocking Ports from Unauthorized Use

Even if you search for servers you don't want running, you might miss something, or a server might start up without your knowledge at some point in the future. For this reason, you may want to set up blocks on the unauthorized use of network ports. Such barriers usually come in the form of a *firewall*, which is a program or computer that functions as a gatekeeper between one or a small group of computers and a wider network.

Some firewalls are designed to protect many computers, and the GunkBuster's Notebook sidebar "NAT Appliances" covers one such type of firewall you might consider purchasing to protect a small network. In terms of protecting a single Linux system with Linux's own tools, though, you'll use the `iptables` tool or a GUI front end to it. This program enables you to create a number of

firewall rules that help you to block unwanted accesses to your computer. Unfortunately, using `iptables` directly requires creating potentially dozens of complex rules. Fortunately, most Linux distributions provide simplified GUI front-ends to `iptables`. For instance, Figure 10-7 shows the Fedora and Red Hat Security Level Configuration tool, which you can find on the System Settings > Security Level menu or launch by typing `system-config-securitylevel` as root in an X-based command prompt window.

A basic firewall provides very limited options. For instance, the Security Level Configuration tool shown in Figure 10-7 enables you to enable access to particular servers by checking their boxes in the Trusted Services area. If a server isn't listed, you can add it by specifying its port number in the Other Ports field. When you set Security Level to Enable Firewall and click OK, the system should close off access to all other ports.



Figure 10-7

Firewall configuration enables you to leave open access to some ports while restricting access to others.

TIP: Other distributions provide firewall tools under other names, such as SuSE's YaST and YaST2. If you can't find a GUI firewall for your distribution, you can download and run a generic tool to create a firewall script, such as Knetfilter (<http://expansa.sns.it/knetfilter/>), Firestarter (<http://www.fs-security.com>), or GuardDog (<http://www.simonzone.com/software/guarddog/>). Alternatively, you could dig into the `iptables` documentation and write your own firewall script without the help of a GUI tool. You may also need to look to a third-party tool or a raw `iptables` script if you need to create more sophisticated firewall rules than is possible with simple tools like Security Level Configuration.

GunkBuster's Notebook: NAT Appliances

If you're running a computer or small network that's connected to the Internet via a broadband link (such as a DSL or cable modem), you may want to look into a standalone network address translation (NAT) appliance, often referred to as a *broadband router* or *broadband firewall*. These devices connect to your ISP just as your computer normally does and then use NAT to enable all of your computers to connect through the device. NAT is a technique that enables one computer to "masquerade" as many. For instance, when a NAT-connected computer tries to contact a Web site, the NAT device intercepts the outgoing connection request and changes the IP address in the outgoing packets. When a response comes, the NAT device reverses the change. Thus, neither the server nor the client knows that a change has occurred.

This extra step has certain advantages. For one thing, the NAT computer consumes just one IP address on the Internet at large, but it can serve theoretically huge numbers of computers—certainly enough for a typical home or small business network with a dozen or so systems. Another advantage of NAT is that it provides security that's similar to that of a firewall. (Although these devices are often marketed as firewalls, they might not technically qualify as such, depending on the definition you're using.) Specifically, incoming requests from the Internet at large terminate at the NAT appliance, so a would-be cracker can't connect to any unprotected ports on your local computers without first breaching the NAT router's security.

NAT routers do have their own drawbacks, though. Although NAT doesn't interfere with most protocols, it does slow down or completely block others. Some online games and security



protocols don't work well through these routers, for instance. Some of these problems can be worked around, but you'll need to do some online digging to find the solutions. (They're often very specific to particular devices.)

Using Encryption

The Internet is not a secure place. Data sent to or from the Internet is likely to pass through a dozen or more computers and might be monitored (or *sniffed*) from still more systems. Thus, when you send sensitive data, such as a password or credit card number, over the Internet, it could be intercepted and fall into unfriendly hands. Even seemingly innocuous data could conceivably be used against you—say, to impersonate your identity in a low-security Web forum.

Fortunately, Internet security has improved greatly over the past decade thanks to the increasing availability of good encryption software. To be effective, though, you must *use* this software. The first step on this road is knowing it exists. Some key types of software and protocols include:

- ✓ **File transfers**—Using the older File Transfer Protocol (FTP) to download files from anonymous sites is okay because you're not using a password and the files are publicly available and so don't contain sensitive data. The same is true for the Hypertext Transfer Protocol (HTTP) used on the Web. When transferring sensitive files between computers on the Internet at large, though, or when you must log in using a unique username and password, use SSH if at all possible. SSH supports encryption and so is much better than FTP when security is important.
- ✓ **Remote logins**—If you need to log into a remote computer to control it, use SSH. The older Telnet protocol, while still widely available, is woefully inadequate for login sessions. The Virtual Network Computing (VNC) protocol encrypts passwords but not other data, so it might be acceptable for low-sensitivity logins or over local networks but not for performing even remotely sensitive tasks on the Internet at large.
- ✓ **Web transactions**—As described in Chapter 5, you should ensure that your Web browser has negotiated a secure HTTP connection rather than a conventional HTTP connection before you send credit card numbers or other sensitive data over the Web. In most cases, Web browsers display a padlock or key icon in the lower-left or lower-right corner of the window when using a secure connection. Note that many sites require passwords but don't negotiate secure connections. This is tolerable for low-security sites (say, for authenticating to an online newspaper), but you should be sure to not use the password for such sites on more secure sites (such as your bank's online banking site).

- ✓ **E-mail**—The Simple Mail Transfer Protocol (SMTP) used by most mail servers is inherently insecure, so you shouldn't consider e-mail to be confidential or send passwords or other sensitive data via e-mail. If you must use e-mail for such exchanges, install encryption software, such as the GNU Privacy Guard (GPG; <http://www.gnupg.org>) on both computers and use it to encrypt your messages.
- ✓ **File sharing**—Protocols to share files as if they were local, such as the Server Message Block/Common Internet File System (SMB/CIFS) used by Windows and Linux's Samba package, are intended for use on relatively secure local networks. Although SMB/CIFS can optionally use encrypted passwords, this encryption can be quite weak and non-password data are exchanged in the clear.

The worst threats concerning nonencrypted data have traditionally been on the Internet at large; because data pass between so many systems, many opportunities exist for traffic to be intercepted. Local networks, by contrast, have been relatively secure because an intruder would need access to a compromised local system or even physical access to the network wires to intercept local network traffic. Thus, security on local file-sharing protocols, to name just one example, has traditionally been relatively low. Similarly, many people have used Telnet, FTP, and other unencrypted protocols locally, even when they wouldn't dream of using these protocols on the Internet at large.

The explosion in popularity of wireless network protocols, though, changes this equation. The default security for wireless network hardware is typically quite low. Although many devices do provide options to encrypt data, users often don't activate these features, and sometimes the wireless encryption can be broken relatively easily. Thus, if you're using a wireless network, you should do two things:

- ✓ Read your manuals and activate any built-in encryption on your wireless networking hardware.
- ✓ Don't assume that your "local" wireless traffic is secure, even if you activate the wireless encryption. Somebody sitting in a car outside your home or office might be able to listen in on your local traffic, plucking passwords or other sensitive data out of the air. Similarly, such intruders might be able to break into your local systems, bypassing security provided by a NAT router.

Monitoring for Intrusion

You should always keep in mind the fact that security is never absolute. No matter how carefully you remove unnecessary servers, block ports with firewalls,

use encryption, and perform other preventive security tasks, there's always the possibility that a cracker will be one step ahead of you. For this reason, you should keep an eye out for any unwelcome guests, just as you remain alert to signs of cockroaches, mice, and other unwelcome creatures in your house.

Fortunately, computer intruders don't leave droppings around your computer. Unfortunately, this fact means you must use other means to locate them. Some ways of monitoring for intruders include the following:

- ✓ **Know your system.** If you're familiar with how your computer normally operates, you're more likely to be able to spot changes caused by an intruder. For instance, a system compromise might result in an increased system load (described in Chapter 8, "Managing Processes") or a previously reliable program behaving unreliably, or simply differently. If you see such changes, don't just shrug them off as part of the mystery of life—investigate them. The degunking procedures described throughout this book will help you track down the source of a change in behavior. If that change seems to be due to new software you didn't install or configuration file changes you didn't make, that could be a sign of an intrusion.
- ✓ **Monitor log files.** Linux stores *log files* in `/var/log` and its subdirectories. These files record details of the operation of servers and certain other critical system processes. Such files frequently contain clues to an intrusion. Unfortunately, reviewing every change in every log file would be a full-time job, so you may need to restrict such reviews to occasions when your suspicions have been aroused because of some other event. The most important log file is usually `/var/log/messages`, but some systems have a special security log file (often `/var/log/secure`). Messages reporting logins you or other authorized users didn't make are a cause for alarm. Missing messages can also be a sign of trouble—intruders often remove entries that show signs of their intrusion. This can leave suspicious gaps in log files. (Entries are usually time-coded.)
- ✓ **Monitor for changes to files.** One advanced intrusion-detection technique is monitoring for changes to system files. Tools such as Tripwire (<http://www.tripwire.org>) can help by keeping checksums of all the important files on your system in an encrypted database. By comparing these checksums to newly created checksums on a regular basis, Tripwire can detect intrusions quite reliably. Unfortunately, proper Tripwire configuration is rather tedious, so I don't describe it in this book. If your computer is exposed directly to the Internet, though, you should seriously consider looking into this software. If you suspect an intrusion and are using an RPM-based distribution, typing `rpm -Va` will verify all the files installed via RPM. Changes to program executable files are highly suspicious, but

changes to configuration files are common and can flood the output. Be aware that this test will fail if the intruder used RPM to install modified software!

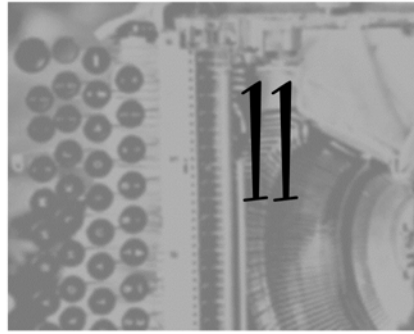
- ✓ **Check for new servers.** Intruders sometimes break in through one server but then install or run a new one to make it easier to return. You can detect these new servers by using `netstat`, as described earlier, in “Removing Unnecessary Servers.” Thus, periodically repeating this review can help you detect crackers.
- ✓ **Check for suspicious accounts.** Just as intruders often add new servers to your system, they often add new accounts or change existing ones. Thus, you should periodically review your account database, as described in Chapter 9, “Account Degunking,” to spot changes in your accounts.
- ✓ **Check for suspicious processes.** Crackers sometimes leave non-server programs running full time, and even when they don’t, whatever they’re running when logged in will show up in a process listing. Thus, periodically checking running processes using `ps`, as described in Chapter 8, may alert you to a problem. On the downside, a typical Linux system is likely to be running dozens of processes, so spotting a suspicious one can be hard. If the intruder didn’t leave something running full time, there’ll also be no hint of the compromise unless the intruder is logged in when you run this test.
- ✓ **Check for root kits.** The `chkrootkit` utility (<http://www.chkrootkit.org>) checks for *root kits*, which are prepackaged intrusion tools used by many crackers. If you suspect your system might have been compromised, you should download the latest version of this tool and run it. You might want to do this on a regular basis if your system is particularly sensitive.

WARNING! *Wily intruders often change system utilities, such as `top`, `ps`, and `netstat`, to hide the signs of their intrusion. Thus, you can’t count on these utilities to be reliable tools for tracking down intruders. To be sure, they do often work, but only because crackers often don’t replace all the utilities you might use to detect them.*

How often you should perform these steps depends on how sensitive your system is. For a typical home system that sits behind a NAT router, running `chkrootkit` and one or two other tests once every few months or if you notice something suspicious should be fine. On a sensitive server that’s constantly exposed to the Internet, installing Tripwire and running some checks once or more a day (probably in an automated way) may be in order. Of course, basic attention to your system’s normal operating characteristics applies to any computer.

Summing Up: Keeping Network Gunk Off of Your Computer

The Internet is a wonderful resource, but it's also become a major source of gunk. Web-based and e-mail gunk are particularly common. Web-based gunk can be fought through options in your Web browser and through the use of a proxy server, which can clean more types of gunk than browser-based rules and serve multiple computers. E-mail gunk is typically fought through the use of spam filters, which can be integrated into a mail server or run from your mail reader. In addition to these forms of gunk, the Internet can be a medium for intruders to invade your system. Protecting your computer by removing unnecessary servers, setting up a firewall, using encryption, and monitoring for intrusion is vitally important if your system is connected to the Internet.



Finding Drivers for Your Hardware

Degunking Checklist:

- ✓ Identify the source of hardware-related problems.
- ✓ Know where to go to find drivers for your hardware.
- ✓ Know how to install drivers in the Linux kernel and in other programs that use drivers.
- ✓ Know how to replace hardware—when it's time to give up trying to get something to work, and how to identify a Linux-compatible replacement.

Degunking hardware is always tricky. Part of the difficulty is identifying the true source of the problem—sometimes what looks like faulty hardware can turn out to be bad drivers. The issue is complicated by the fact that most hardware manufacturers make Windows drivers for their products, but more often than not, they ignore Linux. Thus, an initial hurdle in getting any hardware working with Linux is finding drivers for it. Although Linux ships with drivers for most common hardware, you should know where to find the drivers when you install new hardware or when you install Linux on old hardware. This knowledge alone will solve a lot of problems. Once the drivers are found, of course, you must be able to install them. Finally, hardware problems can sometimes be solved only by replacing the hardware—perhaps the hardware is broken, or perhaps it's simply unsupported under Linux. In either event, before you drop hard-earned cash on new hardware, you should know how to determine that it really does need to be replaced, and you should also know how to determine if new hardware is compatible with Linux.

Hardware Troubleshooting

Gunky hardware usually causes pretty obvious problems. Perhaps the computer keeps crashing, or maybe a printer doesn't deserve the name because it won't print. Unfortunately, symptoms like these won't lead you directly to the cause of the problem. For instance, a flaky printer might be flaky because its connecting cable is inserted loosely, because the Linux kernel lacks drivers to communicate with the device, because your printing subsystem lacks drivers to format output that the printer can understand, because the printing subsystem isn't running, or any of dozens of other reasons. Describing every possible problem and its symptoms would take a book far larger than this one. Instead, I first cover some general hardware troubleshooting techniques. These should help you isolate the problem. I also describe some of the most common hardware problems.

General Hardware Troubleshooting Procedures

Hardware troubleshooting can be particularly onerous because hardware problems tend to be unusually slippery. A component might work fine today but then fail tomorrow, bringing down the computer. Perhaps the worst types of problems are those that occur both randomly and infrequently—say, bad RAM that causes a system crash once a week. This time scale, combined with the unpredictability of the problems, makes it hard to perform tests designed to isolate the cause.

Although hardware troubleshooting can be a difficult task, it's not an impossible one. Some general procedures can help you reduce the amount of hair you pull out in the process:

- ✓ **Take notes.** It's easy to forget what you've tried, or what the outcomes were, when you delve into fixing a hardware problem. Taking notes can help you collect and organize your thoughts.
- ✓ **Be methodical.** Think back to your high school science classes and treat your debugging session like a scientific experiment. Formulate two competing hypotheses about the cause of a problem, develop a test to differentiate between these causes, and implement that test. For instance, the hypotheses might be that a problem is a bad cable or that it's not a bad cable. The test could then be replacing the suspect cable.
- ✓ **Check log files.** Linux stores various log files in the `/var/log` directory. These files may hold useful clues about the cause of the problem. Check the ends of the files soon after a problem occurs, if possible. You might also find some helpful hints in the *kernel ring buffer*, which holds messages from the kernel. Typing `dmesg` at a command prompt displays this buffer.
- ✓ **Do some research.** Try a Web search on keywords related to your problem. This approach is most likely to work if the system delivers an error message; you can enter that message into the search engine. I find that a search of Usenet newsgroups (at <http://groups.google.com>) is often more likely to turn up useful information than a general Web search.
- ✓ **Determine triggering conditions.** Try to find out what triggers the problem. Sometimes this is very hard to do because hardware problems can often be fairly random occurrences. If you notice that a problem occurs when something in particular happens, though, try variants on that trigger event. For instance, if a printer prints documents from the KWord word processor very slowly but prints quickly from OpenOffice.org, try experimenting with KWord print options. You might also try printing from other KDE-based applications. Perhaps you'll find that the problem is with KDE applications generally or is related to a particular printer option. This won't solve the problem completely, but it will help you further isolate the cause.
- ✓ **Try multiple OSs.** One of the trickiest aspects of hardware debugging is figuring out whether the problem is an actual hardware fault or a buggy driver. Checking the hardware in another OS can be a good way to help on this score. Of course, this is easiest if your computer multi-boots, but you could try temporarily installing FreeBSD, Windows, or some other OS (perhaps on another hard disk, if the problem isn't with your main hard disk). Even a FreeDOS (<http://www.freedos.org>) boot floppy can be helpful in some cases. If the problem persists across multiple OSs, it's very likely an actual hardware fault. If it persists in just one OS, chances are it's a driver or configuration issue.

- ✓ **Swap out hardware.** If possible, try replacing the hardware that's giving you problems. This isn't always economically viable, but sometimes a component is inexpensive enough that replacing it on a hunch isn't unreasonable. Other times, you can swap components between computers or even make do with less hardware. (If necessary, enlist the aid of a friend for temporary component transplants.) For instance, if your computer has two memory DIMMs and you suspect one is flaky, you could pull out one, test the system, then swap the DIMMs. If one configuration works and the other doesn't, you've found your bad memory.

These procedures aren't guaranteed to solve your problems, but they should help bring you closer to a solution. Hardware troubleshooting can be frustrating, and it takes diligence, patience, and general problem-solving skills to get to the bottom of most problems.

Common Hardware Problems

Although the number of hardware problems that can manifest themselves in a Linux environment is truly enormous, a few problems are particularly common. These issues crop up time and time again, so checking for them may be worthwhile when you run into problems.

NOTE: One particularly common set of hardware problems relates to X server configuration. This topic is so important, and so broad, that Chapter 12, "Optimizing Your X Configuration," is devoted entirely to it.

Correcting System Crashes

Perhaps the most frustrating type of problem is a system crash—the computer simply stops responding to input and perhaps reboots itself. Linux is normally a very robust OS that crashes very infrequently. Linux systems frequently run for months at a time without reboots or crashes. Thus, if your system is crashing, something is wrong with it, and that thing is almost certainly a hardware or driver issue.

Driver-related crashes can usually be identified because there's a clear triggering event. For instance, if your system hangs when you try to burn a CD-R, it could be that your ATA or SCSI controller's driver is less than stable, or perhaps you're trying to use inappropriate options in your CD-R burning software. (Normally, the latter shouldn't cause a system crash, but in combination with a driver bug, it might.) If your system simply locks up or reboots at random times, hardware faults are likely:

- ✓ **Flaky power supply**—One of the most common sources of problems is one that most people are likely to overlook: the power supply. This is a rectangular box in the computer case, into which you plug the power cable that runs to the wall. Power supplies can be too weak to drive all the hardware in the case, which can cause random failures or problems when you try to access certain devices, such as CD-ROM drives. Power supplies can also fail over time. I recommend buying a well-respected brand, such as PC Power and Cooling or Antec. Cheap generic power supplies are more likely to cause problems.
- ✓ **Bad RAM**—In my experience, bad RAM is among the most common sources of Linux system crashes. Sometimes simply shutting down the computer, pulling the RAM modules out, and re-inserting them will fix the problem. Other times, you must replace the RAM. Be sure to consult the motherboard's manual to see what type of memory to get.
- ✓ **Bad CPU**—Sometimes a CPU goes bad. Such problems are most likely to manifest when you install a new CPU—it might be defective or incompatible with your motherboard. In my experience, working CPUs seldom just go bad. If you've been overclocking your system, though, you might just kill your CPU.
- ✓ **Overheating**—Inadequate ventilation can cause a CPU or RAM to overheat, which can cause system crashes or other flaky behavior. Sometimes adding or replacing a case fan or the fan that attaches to a CPU can do wonders. If you replace a CPU fan, be sure to buy one that's adequate. CPU manufacturers have lists of approved fans for particular CPU models on their Web sites.

Modifying Device Modes

Linux relies heavily on its file ownership and permissions to control access not just to conventional files, but to hardware devices. For instance, ordinary users can't normally destroy a disk partition because the disk partition files (such as `/dev/hda1` and `/dev/sdb5`) are owned by root and only root may write to those files.

Some devices, though, really should be accessible to ordinary users. For instance, in order to play sounds through a sound card, you need access to audio device files, such as `/dev/dsp`. Most Linux distributions, and particularly those intended for desktop use, use one scheme or another to make such files accessible to ordinary users. One common system is to change the ownership of these files to whoever has logged in at the console. Another approach is to create a special group (such as `audio` for sound devices) that has appropriate access rights to the relevant files and place ordinary users in that group. Both of these

approaches work most of the time, but both can fail to work under certain circumstances. For instance, if a user's not in the `audio` group, that user might be unable to access the audio devices. Typically, attempting to do so will result in either a "permission denied" error message or simply a failure to produce sound (or otherwise access hardware).

As a test for this problem, try running the application as `root`. If `root` can use the hardware but an ordinary user can't, the problem is almost certainly a permission issue. The solution depends on how your distribution is controlling access to the device. If the distribution automatically changes device ownership, perhaps logging out and then logging in again will fix the problem. If the distribution uses a group-access approach, adding the user to the appropriate group should do the trick. (You may need to log out and back in again before the change will take effect, though.)

Similar access problems exist with USB devices, but many USB components are controlled through files in the `/proc/bus/usb` directory tree. This directory tree is virtual—that is, it's created on the fly by the USB drivers in the kernel. You can change permissions on these files, but these changes won't persist across a reboot. To make a change persistent, you must alter the Linux hotplug configuration. Unfortunately, this is still a rather unfriendly configuration component, and the details depend on the device you want to add. Broadly speaking, you would edit the `/etc/hotplug/usb.usermap` file to add an entry for your device and then create a script named after that device in `/etc/hotplug/usb`. This script would modify the permissions on the files that correspond to the device. Because these entries and scripts are highly device-specific, you'll need to research them yourself. Try entering keywords like "hotplug" and the name of the device in a Web search engine. With luck, you'll find appropriate configuration files for your hardware.

Speeding Up Slow Disks

Linux usually does a good job of configuring hard disks to perform reasonably well; however, sometimes it doesn't. If your hard disk performance seems sluggish, you should first try to test it using the `hdparm` utility:

```
# hdparm -tT /dev/hdb
```

```
/dev/hdb:
```

```
Timing cached reads:   928 MB in  2.00 seconds = 463.38 MB/sec
```

```
Timing buffered disk reads: 116 MB in  3.03 seconds = 38.24 MB/sec
```

NOTE: This command must be run as root. Ideally, you should run it with as few other programs running as possible. You may also need to run it several times to get an accurate reading—it can be thrown off by unrelated disk activity that happens to occur at the same time you run the program.

The first line of output from this command (cached reads) is a measure of cached disk performance, which means it's effectively a test of the computer's memory performance. The second line (buffered disk reads) is the measure of disk performance. In this case, it shows a speed of 38.24MB/s. New disks should perform at least as well as this (this disk is about a year old as I type and will be older by the time you read this). If you've got old disks, though, their performance could be worse.

One cause of poor disk performance is old hardware—both the disk and its controller or host adapter. Replacing either might help matters, but of course replacing the disk would mean moving Linux to the replacement disk. Before you do anything drastic, though, you might want to investigate `hdparm`'s disk-tuning options. In particular, `-d1` enables DMA mode, which usually improves performance compared to the older PIO mode, and `-X` can be used to set any of several DMA transfer modes. On modern hardware, the following command should optimize disk performance:

```
# hdparm -d1 -X udma6 /dev/hda
```

This command sets UltraDMA mode 6 on `/dev/hda`. On older disks, setting a lower UltraDMA mode (such as UltraDMA mode 4, `udma4`), multi-word DMA modes 1 or 2 (`mdma1` or `mdma2`), or even simple DMA modes 0, 1, or 2 (`sdma0`, `sdma1`, or `sdma2`) may be necessary. If changing the DMA mode significantly improves your disk performance, you can add the command to a system startup script, as described in Chapter 8, “Managing Processes.”

WARNING! Adjusting a hard disk's DMA mode inappropriately can cause the disk to hang. A hung disk can translate to a hung system, so this procedure is very risky. You should attempt it only if you're convinced that your hard disk isn't performing up to its potential, and then you should save any unsaved work and unmount any filesystems that don't need to be mounted as a precaution.

Fixing Flaky Network Connections

Linux provides excellent networking support, and this system usually works reasonably well. Sometimes, though, configuration problems or bad hardware

can result in performance issues. If your network connection seems unusually slow or unreliable, you should begin by checking some basic statistics using `ifconfig`:

```
$ ifconfig eth0
```

```
eth0  Link encap:Ethernet  HWaddr 00:80:C8:FA:3B:0A
       inet addr:172.24.21.5  Bcast:172.24.21.255  Mask:255.255.255.0
       inet6 addr: fe80::280:c8ff:fefa:3b0a/64  Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:346968 errors:0 dropped:0 overruns:0 frame:0
       TX packets:340423 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:74008469 (70.5 MiB)  TX bytes:386741281 (368.8 MiB)
       Interrupt:11 Base address:0xee80
```

This command displays the status of the `eth0` (first Ethernet) device; you may need to change `eth0` to something else if you're not using Ethernet. Omitting the device entirely will also work, but the result will be information on all your network connections, including the loopback (`lo`) interface, which probably isn't important.

In any event, you should first look at the `RX packets` and `TX packets` lines. In particular, look at the number of errors and dropped packets. If these numbers are high relative to the total number of packets, it indicates a problem. Typically, such errors are caused by bad network cables or a flaky network interface card, so try replacing these components. If you replace your network card, try using a well-known brand, such as Intel, 3Com, or Linksys. Many generic brands use the cheapest possible components, which are more likely to cause problems, particularly if the hardware is pushed in a high-demand application.

The number of collisions can also be diagnostic of problems. A collision results when two computers try to send data simultaneously. The result is like two people trying to talk at the same time; neither party can be understood. Collisions generally indicate high load on your network. In particular, if you use hubs rather than switches to connect computers and if your network sees a lot of traffic, collisions are the inevitable result. You may be able to improve performance by replacing your hubs with switches. The switches are smarter about handling the traffic and so can direct it in such a way that collisions are avoided.

Find Sources for Hardware Drivers

Many hardware problems trace their way back to drivers. Thus, you must know something about Linux drivers when degunking your hardware for Linux. At the root of this knowledge is knowing where to look for drivers, in the sense of both where they're installed in a Linux system and where to find new drivers when you need to upgrade your drivers.

Broadly speaking, driver sources fall into three categories: the Linux kernel, third-party kernel drivers, and drivers embedded in non-kernel software. Each type of driver can be sought out in its own way. (The upcoming section "Install Drivers" covers getting your driver working once you've found it.)

The Linux Kernel

The primary source of Linux drivers is the Linux kernel. The kernel is the heart of any OS. It doles out CPU time and memory to programs, manages filesystems, controls network access, and serves as an interface between the hardware and non-kernel software. It's this last role that drivers fill; hence, the kernel is the source of most drivers. (Some drivers are in software components that can talk more directly to hardware than most software can or that talk to hardware that's interfaced to the computer via other hardware that the kernel controls.)

Sources of Kernel Drivers

Linux's open source nature makes for an unusual structure to its kernel. Specifically, early versions of Linux required all drivers to be compiled into the main kernel file. This meant that a Linux kernel was typically customized for the hardware on which it ran—if you needed a driver for a specific SCSI host adapter, you compiled that driver into the kernel. By contrast, other OSs would have the main kernel file load a separate driver file provided by the hardware manufacturer. All modern versions of Linux also support this separate-file model for drivers, but you can usually still compile a driver directly into the main kernel file if you like. Doing so can simplify configuration slightly because you don't need to tell the kernel to look for a kernel module (that is, a separate driver file). Compiling the driver into the kernel also greatly simplifies the boot process when the driver handles the boot device (that is, your hard disk controller). If the kernel needs to load a separate module, it needs to be able to access the module file, but if that's stored on the device that the driver controls, a chicken-and-egg problem ensues. (The typical solution is to put the driver module in a temporary RAM disk, which is an area of RAM that Linux can

treat like a disk drive and that the boot loader can load into memory from the disk using BIOS calls.)

If you've found that a problem is caused by a bug in a kernel driver, you must upgrade that driver to fix the problem. (Sometimes downgrading the driver to an older version will also fix the problem.) Broadly speaking, you have three choices for how to proceed:

- ✓ **Upgrade your distribution's kernel package.** Major Linux distributions provide their kernels in packages, just like other software packages. You can use the package management tools described in Chapter 6, "Cleaning Out Unused Packages," to upgrade your distribution-provided kernel package. This approach tends to be fairly straightforward if you're familiar with Linux package management tools, but kernel upgrades often take some time to filter down through distributions, so you might have to wait a bit to fix the problem.
- ✓ **Locate a package upgrade for the problem driver.** Occasionally, third parties release driver module upgrades for buggy or missing drivers in popular distributions. Such packages are usually intended to fill gaps in the driver coverage provided by the distribution, though, rather than fix buggy drivers.
- ✓ **Recompile the kernel.** You can download the original kernel source, compile it yourself, and use the kernel you compile locally. This approach provides the greatest flexibility but, like any do-it-yourself solution, requires the greatest effort.

Configuring the Kernel

If you decide to recompile the kernel yourself, you should know how to configure it. This information can also be handy in understanding kernel modules generally, so I provide a brief outline of the procedure.

First, you must obtain a copy of the kernel source code. One reliable site for this is <http://www.kernel.org>. Locate and download a complete Linux kernel source tarball—a file with a name of the form `linux-version.tar.gz` or `linux-version.tar.bz2`, where *version* is the version number. Files of the form `patch-version.tar.gz` or `patch-version.tar.bz2` are useless unless you have the immediately preceding version; these files provide just the changes and so are much smaller to distribute to those who keep up with kernel changes in source code form.

NOTE: The Linux kernel version numbers have three parts, as in 2.6.10. The major version number (2 in this example) changes infrequently. The third version number (10 in this example) changes fairly regularly. The middle version number (6 in this example) has a special meaning: Even numbers denote **stable** kernels, which are deemed ready for use by the general public. Odd numbers denote **development** kernels, in which major new features are added. You should avoid development kernels unless you're prepared to deal with less stable operation and really need a feature that's been added to such a kernel but that's not yet available in a stable kernel.

Once you've downloaded the kernel source code, unpack it in a convenient location (the traditional location is `/usr/src`):

```
# tar xvzf /path/to/linux-version.tar.gz
```

In this example, `/path/to` is the directory in which the file resides. If you downloaded a bzip2-compressed file (one with a name that ends in `.bz2`), change `tar xvzf` to `tar xvjf`. The result is a directory called `linux-version` in which the Linux source code resides. You should create a symbolic link to this directory called `linux` by typing `ln -s linux-version linux`.

NOTE: Ordinarily, `root` owns `/usr/src`, so only `root` can extract the kernel source code to this directory. If you change the ownership of this directory, or if you change its permissions to enable others to write to it, an ordinary user may extract the kernel source files, configure the kernel, and build it.

At this point, the Linux kernel source is ready to be configured. Change into the `linux` directory and type **make xconfig**. The system should churn for a while and then present a window similar to the one shown in Figure 11-1. You can configure the kernel from this window, but be prepared to spend several minutes doing so, and perhaps over an hour if you're unfamiliar with the process. The major areas of configuration appear in the pane on the left of the window. Click an area to see that area's options in the top right pane. Most options have a square box to their left. If the box is empty, the feature won't be compiled. If it contains a dot, the option will be compiled as a module. If the box contains a check mark, the option will be compiled directly into the main kernel file. (Some items cannot be compiled as modules or cannot be compiled directly into the main kernel file. This is normal.) When you select an item, a description for it usually appears in the bottom right pane, but some items lack descriptions.

NOTE: If you get an error message rather than the window shown in Figure 11-1, chances are you're missing some development libraries. Typing **make menuconfig** may enable you to configure your kernel using a text-mode tool rather than a GUI tool, or you may need to track down the missing development tools.

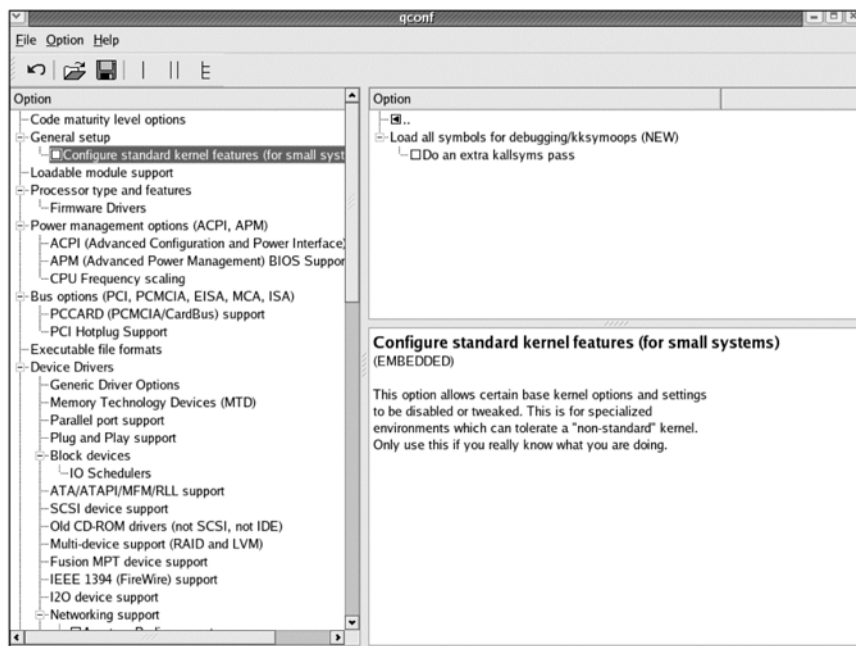


Figure 11-1

Linux kernel configuration is done via a GUI (shown here) or text-mode tool.

The kernel contains far too many options for me to describe them all here. Instead, you should peruse them yourself and read their descriptions. If in doubt, follow the advice provided with the option concerning whether to compile it. When you're done configuring your kernel, select **File > Quit** and click **Save Changes** in the dialog box that results. This saves the changes you've made to your kernel configuration, enabling you to compile the kernel, as described later in "Installing Kernel Drivers."

You should realize that the standard kernel downloaded from "generic" Linux kernel sites is not likely to have a default configuration that matches your distribution's kernel. A stock kernel may lack important drivers—say, for your disk controller. Thus, you'll need to pay careful attention to these options. Chances are you won't understand all of them, and you could easily leave something important out. Thus, kernel recompilation is really an advanced degunking technique. For those who learn to do it, recompiling the kernel can be very useful. By omitting unnecessary drivers and options, you can save space and improve overall kernel efficiency. Recompiling the kernel for your CPU can also improve efficiency. The problem is that getting it right can be very difficult for the uninitiated.

TIP: Even if you don't intend to recompile your kernel, perusing its options in this way can be very informative. You'll learn about kernel options, available drivers, and so on. Perhaps you'll spot something that will prove useful in the future, such as the presence of alternative drivers for a device or drivers for some hardware you're considering buying.

Third-Party Kernel Modules

A second major source of drivers is third-party kernel modules. These are Linux kernel modules that aren't part of the main Linux kernel source tree, as downloaded from sites such as <http://www.kernel.org>. Broadly speaking, these modules fall into two categories:

- ✓ Open source drivers that have not yet been added to the kernel. Perhaps the developers just don't want to have their project added to the kernel, or perhaps they're on a track to be added eventually but they haven't yet been included for one reason or another.
- ✓ Proprietary drivers that are never likely to be added to the kernel. These drivers are sometimes released by hardware manufacturers as a way of supporting Linux. Typically, such drivers come in binary form that work only with particular kernels or as binary files that can be linked to the kernel using separate source code "glue."

Generally speaking, I recommend avoiding third-party kernel modules. The open source variety are often works in progress, meaning that they may prove unreliable—and unreliable kernel modules can be dangerous beasts indeed because they can more easily crash the computer than other types of buggy software. Proprietary drivers can be worse in some respects—you may find that you want or need to upgrade your kernel but the proprietary drivers won't work with the new kernel.

NOTE: Not all Linux drivers available on hardware manufacturers' Web sites are proprietary. In fact, most manufacturers simply make the standard Linux kernel drivers available in this way. Sometimes the manufacturers contribute to the drivers' development, but other times they've just dropped the drivers on their Web page as a sort of billboard proclaiming the Linux compatibility of their product. You shouldn't go to much effort to try to use a manufacturer-provided driver if a standard Linux kernel driver already exists and works. If the standard Linux driver is flaky, though, you might give a manufacturer's driver a try.

Of course, you may not have much choice in the matter of using third-party drivers. The usual reason for seeking them out is that a feature is missing or unreliable in the standard Linux kernel. If you already own the hardware, you

might as well try the third-party driver if it's your only choice. In most cases, this solution is less expensive and simpler than replacing the hardware.

Some third-party drivers come in source code form. This is particularly true of open source drivers. Typically, you must compile the driver following its instructions, whereupon it becomes available as a kernel module. Sometimes you must *patch* your existing kernel, which means merging the third-party code with your kernel and then recompiling your kernel. Follow the instructions provided with the driver if this is the case.

Some third-party open source modules and many proprietary drivers come in binary form, either as tarballs or in packages you can install using the Linux package tools described in Chapter 6. Again, you should consult the driver's documentation for details, because the installation procedures vary greatly from one driver to another.

Hardware-Specific Software

Some types of hardware require non-kernel drivers. Examples include video cards, printers, and scanners. With the exception of video cards, these devices usually interface to Linux through a kernel-controlled device, such as a USB port or a parallel port. The end result is that the kernel controls the communication with the device but a non-kernel application controls the device at a higher level, such as sending commands to cause a scanner to scan a document.

Because these devices require non-kernel drivers for full functionality, you must locate those drivers elsewhere. As with the kernel, though, the Linux applications that control these devices typically come with a wide range of drivers. Hardware manufacturers might or might not provide drivers of their own, and if they do, those drivers might or might not be the standard Linux programs' drivers. The following list includes sources for drivers:

- ✓ **X server drivers**—The major Linux X server is X.org-X11 (<http://www.x.org>). This server rapidly replaced XFree86 (<http://www.xfree86.org>) as the main Linux X server in 2004, but both are usable on Linux, and both ship with a wide variety of drivers.
- ✓ **Printer drivers**—Linux printer drivers are contained within Ghostscript (<http://www.cs.wisc.edu/~ghost/>), which is Linux's PostScript-to-printer translation software, as described in Chapter 7, "Improving Software Performance." You can also add drivers to Ghostscript, and several printer

driver collections have cropped up, such as GIMP-Print (<http://gimp-print.sourceforge.net>) and Foomatic (<http://www.linuxprinting.org/foomatic.html>). These are usually installed along with your printing system, but if your driver selection list seems suspiciously short, you may want to look into adding one or more of these packages.

- ✓ **Scanner drivers**—Linux uses the Scanner Access Now Easy (SANE; <http://www.sane-project.org>) package to control scanners. This package includes drivers (referred to as *backends* in the SANE documentation) to control scanners, so chances are you won't need to look further than the standard SANE driver set.

In all three cases, manufacturer-supplied and third-party drivers are sometimes available. (The printer driver packages are really third-party open source drivers, but most Linux distributions now include at least one of them as standard equipment, so they blur the line a bit.) The same caveats apply to commercial application-based drivers as to commercial kernel drivers. Most importantly, if you use them, you may find it difficult to upgrade the software package that relies on them. Nonetheless, these drivers can be very handy, or even a downright necessity to use some hardware. This is particularly true of new video cards. A few video card manufacturers, such as nVidia (<http://www.nvidia.com>), are now doing a good job of providing Linux drivers that take full advantage of their hardware soon after it's released. The open source drivers provided with X.org-X11 or XFree86 typically lag behind a bit, because the developers need to acquire the hardware and figure out how it works before they can produce drivers.

Install Drivers

The actions you take to install drivers vary depending on the specific driver. In all cases, you should consult the documentation that came with the driver (typically a README file or something similar). Certain commonalities exist for each class of driver, though, so I describe them here, for kernel drivers, X drivers, printer drivers, and scanner drivers.

Installing Kernel Drivers

Most Linux drivers are kernel drivers, and the methods for installing and using new kernel drivers are the most complex and tricky of any driver installation in Linux. The prototypical new driver installation involves configuring a new kernel version, as described earlier, recompiling the kernel, installing the kernel modules, and rebooting with the new kernel. Sometimes, though, the procedure

is simpler: If you have a precompiled driver module for your current kernel, you should be able to use it without reconfiguring your kernel or rebooting the system. (Skip ahead to the GunkBuster's Notebook sidebar "Using Precompiled Kernel Modules" if you've got such modules and want to use them.)

Recompiling a Kernel

The bulk of the work in installing a new driver that's part of the standard Linux kernel is in reconfiguring the kernel, as described earlier, in "The Linux Kernel." Once that's done, you can recompile your kernel by typing a single command: **make**. You must type this command in the main kernel source directory (typically `/usr/src/linux`). You can compile the kernel as an ordinary user, but only if that user owns all the files and directories in the kernel source tree.

The kernel compilation process will take anywhere from a few minutes to over an hour, depending on the options you've selected and your computer's speed. During this time, you should see summary lines appear on your screen as the kernel compilation process proceeds. When it's done, you'll find a new kernel, called `bzImage`, in the `/usr/src/linux/arch/i386/boot` directory.

NOTE: On non-x86 systems, the kernel name may be `vmlinux`, `vmlinuxz`, or something else, and the exact directory location will be different—in particular, `i386` will be replaced by a code for the system's CPU, such as `ppc` for PowerPC systems. Some architectures also use additional subdirectories, so you may need to go digging around a bit to find the kernel file.

You should copy your new kernel file to the `/boot` directory, typically with a name to distinguish it from other kernels. Appending the kernel version number usually works well. For instance, you might type `cp arch/i386/boot/bzImage/boot/bzImage-2.6.10` from the `/usr/src/linux` directory to copy a 2.6.10 kernel to `/boot`. You must be root to execute this command, even if you compiled the kernel as an ordinary user. This command moves the kernel to a standard location, but it doesn't make the kernel usable. For that, you must still install the kernel modules and prepare your system to use the new kernel.

WARNING! Do not overwrite any existing kernel files with your new kernel! If you do so, and if you erred in your kernel configuration, the result will be that you'll be unable to boot your computer. Although it's possible to recover from this error, doing so requires using an emergency Linux boot medium to reinstall a known good kernel and reconfigure your system to use it. This is a rather advanced and messy degunking task, so you're far better off not risking having to do it.

Installing Modules

To install kernel modules, type `make modules_install` as root in the Linux source directory. This command copies the kernel modules to a subdirectory of `/lib/modules` and performs some housekeeping tasks to make the modules accessible.

NOTE: *If you're using a pre-2.6.x kernel, you must type `make modules` before typing `make modules_install`. This extra command builds the kernel modules separately from the main kernel file. With the 2.6.x kernel, Linux changed the basic kernel compilation so that it builds both the main kernel and the modules, so this extra step isn't necessary.*

Booting a New Kernel

Aside from configuring the kernel, the trickiest part of the job of installing a new kernel is configuring your system to boot with it. Booting a kernel is handled by a *boot loader*, which is a program that the computer's BIOS loads. The boot loader is tasked with locating an OS kernel, loading it into memory, and running it. Advanced boot loaders, including those most commonly used with Linux, also permit you to select the OS or kernel you want to use. Thus, the key to booting a new Linux kernel is in telling your boot loader about that kernel.

Two boot loaders are commonly used with Linux: the Grand Unified Boot Loader (GRUB) and the Linux Loader (LILO). LILO is the older of these two products and is configured through `/etc/lilo.conf`. GRUB is newer than LILO and has been slowly replacing LILO as the default boot loader for most distributions, but LILO is still popular. GRUB is configured through either `/boot/grub/grub.conf` or `/boot/grub/menu.lst`. To determine which boot loader your system uses, check for these three files. Chances are only one will be present. If multiple files exist, though, use your package manager to verify the installation of a `grub` or `lilo` package. If both packages are installed, you can't tell which boot loader you're using. In that case, I suggest first configuring your system for GRUB. If that works, uninstall the `lilo` package (it's gunk). If the GRUB configuration doesn't work, try configuring LILO. If that works, uninstall the `grub` package, because it's gunk.

For both GRUB and LILO, you should load the configuration file into an editor as root. The file should resemble Listing 11-1 or Listing 11-2, which show typical GRUB and LILO configurations, respectively. You needn't be concerned with most of the options in these files. You should know, though, that the files come in two main parts: global configuration options and *stanzas* (sections) for each OS or kernel to be booted. The kernel or OS stanzas typically,

but not always, begin with an unindented line (`title` in GRUB, `image=` or `other=` in LILO) and continue with indented lines until the next stanza or the end of the file. This practice helps you identify different stanzas at a glance. To tell the boot loader about a new kernel, you must create a new stanza for it.

Listing 11-1 A typical GRUB Configuration File

```
default=0
timeout=10
splashimage=(hd0,4)/grub/splash.xpm.gz
title Linux
    root (hd0,5)
    kernel /vmlinuz ro root=/dev/hda9
title Windows XP
    rootnoverify (hd0,1)
    chainloader +1
boot
```

Listing 11-2 A Typical LILO Configuration File

```
prompt
timeout=100
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
default=linux
lba32
image=/boot/vmlinuz
    label=linux
    root=/dev/hda9
    read-only
other=/dev/hda2
    label=windows
```

To create an entry for a new kernel, follow these steps:

1. As root, load the configuration file into your favorite text editor.

2. Copy an existing Linux stanza. Be sure it's a stanza for booting Linux, not for booting another OS. In LILO, Linux stanzas always begin with `image=`. In both GRUB and LILO, they refer to a Linux kernel file, typically called `mlinuz`, `bzImage`, or some variant of one of these names.
3. Change the title label for the stanza. In GRUB, change the `title` line; for instance, change it from `Linux` to `Linux with 2.6.10 kernel` if you've installed a 2.6.10 kernel. In LILO, change the `label=` line; for instance, change it from `label=linux` to `label=linux2610`. (LILO labels must be single words, unlike GRUB titles.)
4. Change the copied stanza to refer to the new kernel. In GRUB, the kernel is referenced in the `kernel` line; in LILO, it's the `image=` line that begins the stanza. Typically, you'll change `mlinuz` (the default kernel name in most distributions), `bzImage` (a common name for locally compiled kernels), or a variant to the name of your kernel. In LILO, you must include the complete path to the kernel, as in `image=/boot/bzImage-2.6.10`. In GRUB, you omit the leading `/boot` if and only if `/boot` is a separate partition. If `/boot` is a normal directory on the root partition, you include this reference.
5. Save the changes to the configuration file and exit from the editor.
6. If you're using LILO, type **lilo** as root. This action re-installs the updated boot loader code where the BIOS can find it. No equivalent action is necessary with GRUB because GRUB reads its configuration file during the boot process.

At this point, the new kernel is installed and ready to be used. Because it's the kernel, though, you must reboot the computer to use it. Most distributions provide a shutdown or reboot option as part of the GUI login screen, so you should log out and use it. If yours doesn't provide such an option, or if you're not running in a GUI, type **shutdown -r now** at a root command prompt.

When the system reboots, you should see your new kernel option appear in the GRUB or LILO menu. (With some LILO configurations, you must type the label name directly. Pressing the Tab key at the `boot:` prompt produces a list of labels.) If you don't see your new kernel, perhaps you're using a boot loader other than the one you modified. Another possibility is that you're using LILO but forgot to type **lilo** after making your changes.

If all goes well, you should be able to select your new kernel and boot the computer. At this point, the system should be using the new driver. If you compiled the driver as a module, it might be loaded automatically, or you might need to issue module-loading commands, as described in the GunkBuster's Notebook sidebar "Using Precompiled Kernel Modules."

GunkBuster's Notebook: Using Precompiled Kernel Modules

If you've obtained a precompiled third-party driver module, you can probably forgo recompiling the kernel to use it. Typically, such drivers come either as packages that you can install using package tools, as described in Chapter 6, or with special installation scripts that place the driver modules in the appropriate driver directory. In either event, you should read the documentation that came with the driver to learn how to install it.

Once the driver is installed, you may need to load it to use it. This is often handled automatically, but if not, you can use the `insmod` command to load a kernel module. Type this command followed by the module's name, as in `insmod floppy`. (This example loads the standard kernel floppy disk driver.) If you find that a driver isn't loading automatically when you boot but you want it to do so, you have two choices:

- ✓ Modify a local startup script (as described in Chapter 8) to run `insmod` to load the module. This approach is convenient if you understand the startup script process for your system, but it's considered something of a jury-rigged method.
- ✓ Modify the `/etc/modules.conf` file for your module. (In some distributions, such as Debian and Gentoo, you'd modify a file in the `/etc/modules.d` subdirectory and then run a special command rather than directly modify `/etc/modules.conf`.) Typically, you'll create an `alias` line that points to the module and links it to a specific subsystem, as in `alias eth0 via-rhine`, which maps the `via-rhine` driver to the `eth0` (first Ethernet) device. This method is the preferred way of doing the job, but it requires you to know the name of the device class to which you're linking the device. If a precompiled binary module requires such changes, its documentation should describe them in detail.

Installing X Drivers

To install and use an X driver, you must do two things:

1. Install the driver in the X driver directory. This is usually `/usr/X11R6/lib/modules/drivers` or `/usr/lib/modules/drivers`. If you've downloaded a driver from a manufacturer's Web site or some other source, it may come with an installation script or instructions; if so, follow them. If you merely need to change from one standard driver to another, you can skip this step.

2. Reconfigure X to use the new driver. This is done by editing the X configuration file, as described shortly.

The bulk of the process is actually in step 2. The X configuration file's name varies depending on the version of X your system is running. Most Linux distributions released since 2004 come with X.org-X11, which uses `/etc/x11/xorg.conf`. A few new distributions and most older ones ship with XFree86, which uses a file called `XF86Config` or `XF86Config-4`, which is stored in `/etc/x11` or `/etc`. All of these files have the same format, so the procedure for changing drivers is identical for all of them.

NOTE: *Very old Linux distributions shipped with XFree86 3.3.6 or earlier. This version of XFree86 used a different driver model than XFree86 4.x and X.org-X11. Specifically, XFree86 3.3.6 and earlier used separate X server programs for each class of video card; the X server and its driver were inseparable. XFree86 4.x and X.org-X11 both use a modular approach, in which the X server loads a driver from a separate module file. Thus, if you're using a very old Linux distribution, the instructions provided here don't apply.*

To change the configured X driver, load the X configuration file into an editor as root and look for lines like these:

```
Section "Device"
    Identifier   "Videocard0"
    Driver       "r128"
    VendorName   "ATI"
    BoardName    "ATI Rage 128"
EndSection
```

This section defines the device driver for the video card. The line that's really important is the `Driver` line, which defines the driver that's to be loaded. In practice, X adds `_drv.o` to the value you provide; for instance, this section loads the `r128_drv.o` module file from the X driver directory. You shouldn't change the `Identifier` line at all (it's used elsewhere to refer to the video card). The `VendorName` and `BoardName` lines are intended for human identification. You should probably change them to match the manufacturer and model number of your video card, just for your own future reference.

If you know the name of your driver module, change the `Driver` line to match it, stripping away the `_drv.o` part of the filename. If you don't know the name of your driver file, your task is more difficult. You could examine the files in the driver directory to see if one strikes you as correct. For instance, if you know you've got a Trident video chipset, the `trident_drv.o` file might jump out at

you and you'd try `trident` on the `Driver` line. Another option is to use `X` itself to probe your video hardware. To do so, type `X -config` as root when `X` is *not* running. The result is a file called `/root/xorg.conf.new` or `/root/XF86Config.new`, which holds a sample configuration file. You shouldn't try to replace your working `X` configuration file with this one, but you should be able to locate the video driver from its equivalent of the video card configuration section.

NOTE: To shut down `X`, type either `telinit 3` or `/etc/init.d/xdm stop` as root; which command works depends on your distributions. In some cases, you may need to replace `xdm` with `gdm` or `kdm` in the second command.

Once you've reconfigured `X`, you should test your changes. If `X` isn't running at all, type `startx` as an ordinary user. This should start `X` (although perhaps not in your normal desktop environment). If `X` is running, log out of your `X` session. On most distributions, this act restarts `X`. If it doesn't, look for an option to restart it in the login screen. You could also shut down `X`, as just described, and then use `startx` to test `X`. As a last resort, reboot the computer to restart `X`.

Installing Printer Drivers

Printer drivers ship with the main Ghostscript executable and as add-on driver packs, such as the GIMP-Print (<http://gimp-print.sourceforge.net>) and Foomatic (<http://www.linuxprinting.org/foomatic.html>) collections. Installing these driver packages is usually a matter of using your package maintenance tools, as described in Chapter 6. If you need to use something less common, such as a driver provided by a printer manufacturer, you should follow the instructions provided with the driver.

In any event, Chapter 7 covers the mechanics of configuring a printer to use a particular driver. Once you've installed a driver package, its drivers should appear in the Common Unix Printing System (CUPS) configuration tool, as described in Chapter 7.

Installing Scanner Drivers

Scanner drivers usually come with SANE, as described earlier. To begin, you should install your distribution's SANE package. (You might need to install more than one actual package. Many distributions split up the core SANE package from its frontends and drivers. Check your available packages to determine what you need to install.) If you've tracked down non-standard SANE drivers, follow the directions included with them to install the drivers.

You may need to install a package using your package manager, run an installation program, or copy files to a driver directory.

No matter how you install the drivers, you activate them by editing the `/etc/sane.d/dll.conf` file. This file consists of a series of lines, each of which refers to a single SANE driver. These lines may be commented out by use of a hash mark (`#`) at the start of the line. If you want to use a driver and it's commented out, remove the hash mark and save the file.

TIP: *Some Linux distributions ship with all the drivers in `/etc/sane.d/dll.conf` active. Others ship with most or all of the drivers commented out. Scanner programs will start up more quickly if they don't need to probe for scanners that aren't present, so you should comment out all of the lines in this file except for the ones that refer to scanners you're actually using.*

Once you've uncommented your scanner's driver from `/etc/sane.d/dll.conf`, you should probe for the scanner using `scanimage`:

```
$ scanimage -L
```

```
device 'epkowa:libusb:005:004' is a Epson Stylus Photo RX500/RX510
flatbed scanner
```

The `-L` option probes for scanners using all of the active drivers and reports the result. In this case, one scanner was found, an Epson Stylus Photo RX500 multifunction scanner/printer. If this procedure detects a scanner, you should be able to use scanner applications, such as XSane (<http://www.xsane.org>) or Kooka (<http://www.kde.org/apps/kooka/>), to use the scanner.

Sometimes typing `scanimage -L` doesn't detect the scanner. The first thing to check if this happens is device permissions, as described earlier, in "Modifying Device Modes." Try typing the command as `root`. If you can detect your scanner as `root` but not as an ordinary user, then your device permissions are set incorrectly.

Another likely source of problems is scanner-specific configuration files, which reside in `/etc/sane.d`. These files are named after the scanners they control or their drivers, such as `epkowa.conf` for the driver Epson supplies for its scanners or `umax.conf` for most Umax scanners. These files contain highly device-specific configuration options. In most cases, these options work fine, but occasionally you might need to tweak them. For instance, you might need to change a device filename if you've connected the scanner in an unusual way. If you don't understand the contents of the file, I recommend that you not try to

adjust it without researching it. Try a Web search on “Linux” and your scanner’s make or model number.

Use the Last Resort: Replace Hardware

Sometimes hardware gunk is just too deeply baked on to be removed. When this happens, you must replace the hardware (or do without the hardware). Knowing when to do this and when to hold out for more thorough degunking will help you balance your degunking time against your wallet. If you decide to replace hardware, doing some research beforehand will help you avoid further problems in configuring the replacement—after all, you don’t want to spend much time degunking brand-new hardware!

When to Replace Hardware

Deciding to replace hardware can be a trying experience. Personally, I find it hard to replace hardware that isn’t broken but that’s incompatible with my software; after all, it’s still good! Of course, the meaning of “good” is relative; if hardware isn’t compatible with Linux, it’s not very good if Linux is your primary OS.

As a practical matter, you should perform at least minimal degunking before replacing hardware. Here are some questions you should ask that will help you determine when to give up on degunking and buy replacement hardware:

- ✓ **What is the source of the problem?** If the hardware itself is defective, you have little choice but to replace it. (Sadly, it’s uneconomical to repair most computer components.) If the problem is in software, you might be able to find a software solution.
- ✓ **What is the official status of Linux driver support?** If the Linux drivers for the device *should* work, according to all you’ve read, then investing some time in finding a software solution might make sense. Other times, though, Linux drivers simply don’t exist or are so new that they aren’t likely to be usable for months. In such cases, your only viable choice is to replace the hardware—unless you want to develop a driver yourself!
- ✓ **How certain are you that the problem is driver-related?** If you’re absolutely convinced that the driver is responsible for your problems, replacing the device with one that uses another driver can be a viable solution. If the problem isn’t related to the driver, though, you might run into the same problem with a replacement. For instance, if a printing problem is caused by a bad printer cable, replacing the printer but not the cable won’t solve the problem.

- ✓ **Will a replacement fare any better?** Although Linux supports at least some devices from most classes, it's possible that your device type is simply so exotic that you won't find anything with better support. Some problems might also be related to non-driver issues, such as lack of features in the software you're using. (In this case, of course, you should look for replacement Linux applications rather than new hardware.) Thus, before replacing hardware, you should ensure that the replacement will work. (The next section covers this issue in more detail.)
- ✓ **Will replacing the hardware provide extra benefits?** Sometimes replacement hardware will improve system performance in some way, such as by improving speed or providing extra features. In such cases, you might want to give up early on your degunking efforts.
- ✓ **How expensive is a replacement?** Unless you're Bill Gates (in which case, why are you reading a book on Linux?), the cost of a replacement device is likely to be a factor.
- ✓ **How much effort will replacing the hardware require?** Replacing some components, such as mice, is almost trivially easy. Others, such as motherboards, are much trickier to replace.

TIP: Most built-in motherboard components, such as hard disk controllers, can be replaced by adding cards to the computer and disabling the on-board device. This can be a handy shortcut if an on-motherboard device goes bad or is incompatible with Linux.

Of course, some of these questions require that you do at least minimal degunking first. Some require you to research the current market for the device type. All of this is work, and just as with cleaning a grimy old pan, you have to decide when the work of cleaning it up exceeds the work and cost of buying a replacement.

Identifying Linux-Compatible Hardware

If you've decided your current hardware is unsalvageable (or that you don't want to put any more effort into trying to salvage it), the question becomes: How do you find a replacement that will work better? First, you should be aware that some types of devices are supported better or worse than others. Table 11-1 breaks common devices into three categories: those with essentially universal Linux support, those with good but not quite universal Linux support, and those with weak support that should be carefully researched before purchase.

Table 11-1 Devices Requiring Varying Levels of Pre-Purchase Research

Universal Linux Support (Research Minimally)	Good Linux Support (Research Modestly)	Iffy Linux Support (Research Carefully)
CPUs	motherboards	software modems (aka <i>Winmodems</i>)
case and CPU fans	USB controllers	
RAM	IEEE-1394 controllers	USB modems
ATA and SCSI hard disks, internal floppy disks, and tape drives	SCSI host adapters	internal DSL modems
	ATA controllers	oddball USB devices
	external (USB or IEEE-1394)	scientific or specialized
CD-ROM and DVD-ROM drives (including recording variants)	hard disks and removable disks	data input devices
	network adapters	
	video cards	
keyboards (Some models have extra keys that don't work under Linux, though.)	sound cards	
	TV tuner cards	
	printers	
mice	scanners	
monitors	digital cameras	
speakers		
microphones		
Ethernet-based DSL and cable modems		
broadband routers		
computer cases		
power supplies		
cables		

The best way to check for Linux compatibility is to do a bit of research on the Internet. Some resources you should consult include:

- ✓ **Your distribution's hardware compatibility list**—Most Linux distributions provide hardware compatibility lists on their Web sites. These lists identify hardware that's known to work with the distribution's standard configuration. In theory, the lists of any two Linux distributions should be nearly identical, but in practice they differ because of differing testing priorities and support policies. Frequently, hardware that works fine isn't listed simply because it's not been tested.
- ✓ **The Linux Hardware Compatibility HOWTO**—This document, located at <http://www.tldp.org/HOWTO/Hardware-HOWTO/>, attempts to summarize hardware that does and does not work with Linux and to provide pointers to appropriate drivers.

- ✓ **Sites for Linux hardware device classes**—Some sites exist that hold information on Linux support for particular types of hardware devices. Notable among these are the Linux Printing site (<http://www.linuxprinting.org>), the main SANE site (<http://www.sane-project.org>), the X.org-X11 page (<http://www.x.org>), the Linmodems page (<http://www.linmodems.org>), and the Video for Linux site (<http://www.exploits.org/v4l/>).
- ✓ **Hardware manufacturers' sites**—Although not all manufacturers explicitly support Linux, some do, or at least provide links to and information about standard Linux drivers.

You can either research the device category first and then go to the store with a list in hand of devices that should work or go to the store to take notes on available hardware and then research their Linux compatibility. In fact, you might do several rounds of this, going back and forth to pick up more information in each location. This is easier if you're shopping on the Internet—you can do it all from the comfort of your chair. Of course, if you prefer to buy locally, you can do so, and this approach has its advantages, such as easier returns if the item proves to be defective.

By doing your homework first, you shouldn't have any surprises when you try to hook up the device and use it under Linux. In some cases, you'll need to install drivers, but frequently the drivers that ship with Linux will work fine. Many distributions provide hardware-detection tools that run at system boot time, so certain types of new hardware will be detected and configured automatically.

Summing Up: Degunking the Hardware/Software Interface

Hardware can be difficult to degunk because its problems are often hard to track down. A few hardware troubleshooting tips will help you do so, though. Many hardware problems are actually driver problems, so being able to locate new drivers and install them can be very important when solving such problems. Other hardware issues are really in the hardware itself, and in such cases, the usual solution is to replace the hardware. When you do so, you should be sure to research Linux compatibility lest you get hardware that doesn't work any better than the device it's replacing.



Optimizing Your X Configuration

Degunking Checklist:

- ✓ Understand X's operational model so that you can take advantage of its strengths and avoid its weaknesses.
- ✓ Use an appropriate X driver for your system.
- ✓ Set the default X color depth and resolution to obtain good performance and appearance.
- ✓ Enable 3D optimizations for 3D games and applications.
- ✓ Configure fonts so that they look good and display quickly.

If you're using Linux as a desktop operating system (OS), chances are you're interacting with the X Window System, or X for short. X provides the low-level tools of a graphical user interface (GUI), but it's a bit odd compared to the GUIs of other OSs, such as Windows and Mac OS. Understanding these differences will help you deal with X and configure it optimally for your system. This configuration begins with picking the best driver to handle the system. Beyond driver configuration comes X configuration options related to color depth, screen size, and 3D optimizations. Finally, a large part of X degunking relates to fonts. X's font handling systems are complex and potentially confusing, so they can easily collect gunk. Cleaning up X fonts will help the system perform smoothly and look good.

The Role of X

The Unix philosophy, to which Linux adheres, holds that it's better to perform a task by gluing together several small tools than by creating a single monolithic one. If you use several small programs, chances are you can recycle one or more of them in other roles. Using small tools also enables you to swap out just one of those tools for another one if you don't like it. The Linux GUI subsystem follows this philosophy by breaking itself down into several components:

- ✓ **X**—X itself is at the root of the Linux GUI subsystem. X interfaces with the display hardware, sets up a video mode, and displays low-level constructs such as windows, lines, and text. X does not, by itself, provide the tools needed to generate menus, buttons, dialog boxes, or other more complex GUI features.
- ✓ **The window manager**—The window manager provides decorative and functional borders around windows and manages the desktop. Users can select which of several window managers to use.
- ✓ **Widget sets**—In order to create dialog boxes, menus, and so on, programmers rely on tools known as *widget sets*. These programming libraries help create GUI niceties. Each programmer selects a widget set to be used by a program, and users can't easily change this feature. (This is why Linux programs' GUIs aren't as consistent across applications as are the GUIs in most other OSs—widget sets differ in their appearance and operational details.)
- ✓ **Desktop environments**—Desktop environments take window managers one step further. They include window managers and additional support programs to make complete user environments.

Of these four components, window managers and desktop environments are covered elsewhere in this book—specifically, in Chapter 4, “Degunking Your Desktop Environment.” Widget sets aren’t easily swapped out by users, so I don’t provide a chapter on degunking them, although some desktop environment settings actually configure the widget sets they use. X itself, though, can be optimized in various ways, and this chapter describes this task.

Another important feature of X is that it’s a component atop Linux. That is, you can boot Linux and not run X. In fact, you can completely uninstall X from the system, which can make the computer run more quickly because it needn’t provide for X’s needs. (Try that with Windows!) Many Linux server systems run in this way, but desktop systems usually don’t. The fact that X is distinct from the rest of Linux can be important because that distinction helps provide ways to tie into X to use it in unusual ways. One of the most extreme of these unusual X configurations is the fact that X needn’t run on the same computer as the programs it runs.

X is a network-enabled GUI. That is, X-based programs treat X as a network resource, much like a file or print server. Instead of being a resource for storing files or printing documents, though, X provides user interface tools. A program uses network protocols to tell X to display a window, place text in the window, and so on. The result is a rather confusing bit of terminology: The user sits at the X server. The programs the user runs (which may reside on a remote computer) are the X clients. Most people think of servers as being distant computers (or the programs they run) and clients as being the programs run on their local computers, and this is usually correct. X turns this upside down, though. Try to think of it from the program’s point of view. People and the X servers they use are, to a program, just network input/output devices, just like any other server.

NOTE: *Although you can use X to run programs on remote systems, I don’t describe precisely how to do so in this book. X’s server nature shows up in terminology and keeping it in mind can help you understand some of X’s peculiarities, which is why I’ve described it here.*

Because X is a modular component of Linux, you can swap out your X server for another one, and theoretically all of your programs will continue to work. Most Linux distributions released in 2004 and later use the X.org-X11 X server (<http://www.x.org>). This server is derived from XFree86 (<http://www.xfree86.org>), which was the preferred X server through 2004. The shift occurred because of a small licensing change in XFree86, to which many Linux distribution maintainers objected. As of XFree86 4.4 and X.org-X11 6.8 (the

latter is actually the first version of X.org-X11), the differences between these two X servers are tiny. In addition to these open source X servers, the commercial Accelerated-X (<http://www.xig.com>) is also available, but most Linux users are happy with the open source X.org-X11 or XFree86.

Pick the Right X Driver

The first X degunking task you might want to undertake is selecting the best X driver for your system. Chapter 11 describes this job, with an emphasis on picking a driver that works. The two main open source X server packages both provide drivers for the most popular video cards on the market as well as for dozens of older legacy cards.

Although it's not often used on x86 systems, another option is to use a *framebuffer* driver, which goes by the driver name `fbdev`. This driver works in conjunction with a kernel-level video driver of the same name. The advantage of a framebuffer driver is that you can use a single user interface between X and the kernel, no matter what video hardware is installed. Using a framebuffer driver has two main drawbacks. First, there are far fewer kernel framebuffer drivers than there are X.org-X11 or XFree86 drivers, so you're less likely to get good performance. (The kernel is likely to fall back on a generic VESA driver that works with most cards in a low-performance mode if nothing else works.) Second, even at their best, framebuffer drivers typically don't perform as well as direct drivers in X.

X also supports its own VESA drivers (under the driver name `vesa`) and the even lower-level VGA drivers (with a driver name `vga`). These drivers work with almost all video cards, but they're slow. Most video chipsets provide advanced features to help display lines, rectangles, and other geometric shapes quickly, and X drivers for particular video card chipsets can take advantage of these features. The VESA and VGA drivers cannot use these features, though, because they're not part of the VESA and VGA standards.

The bottom line is that you should check your X configuration file (`xorg.conf`, `XF86Config`, or `XF86Config-4`, in the `/etc/X11` or `/etc` directory). Look for a `Device` section for your video card, as described in Chapter 11, and ensure that it does *not* reference the `fbdev`, `vesa`, or `vga` driver. If it does, you may want to change that configuration so that it uses a direct driver for your video card. Chapter 11 describes some ways to locate that driver.

Improve X Performance and Appearance

When you installed Linux, chances are you set various options related to your X configuration. These options have implications for performance and the overall appearance of your system. Thus, you might want to review these settings and tweak them for your system. These options are the color depth of the display, the screen's resolution, the monitor's refresh rate, and 3D optimizations for your video card.

NOTE: Desktop environments now enable you to set the color depth and resolution from their configuration tools, as described in Chapter 4. Such changes affect just your own X session, though. The method I describe here alters the defaults for all users and all sessions.

Setting the Color Depth

Computers use numbers to describe everything, and video display colors are no different. Every pixel on your computer display is represented as a number, and these numbers have certain ranges. For instance, the computer might use numbers from 0 to 255 to describe colors, or 0 to 65,535. Obviously, when more numbers are available for representing colors, more colors will be visible. More visible colors result in better-looking images, particularly if you use the computer to edit or display photographs or run programs that use icons with many colors in them.

What Color Depths Mean

The range of numbers available for representing colors is referred to as the *color depth*. It's usually described as a number of bits, as in an 8-bit display or a 24-bit display. Each bit is a binary (base 2) digit, so an 8-bit display uses binary numbers that are eight digits long. This results in 2^8 , or 256, possible colors. Table 12-1 summarizes the most common bit depths used by X.

Table 12-1 Common Color Depths and the Number of Colors They Support

Depth	Number of Colors
1	2
8	256
15	32,768
16	65,536
24	16,777,216
32	4,294,967,296

Based on this information, you might think that cranking your display up to the maximum possible bit depth (32 in most cases) is the way to go. Unfortunately, a high bit depth has two problems:

- ✓ Storing display information requires RAM on the video card, so higher bit depths require more RAM.
- ✓ Manipulating display information takes time, so higher bit depths slow down video displays.

The first of these problems isn't a big one for modern video cards. A very high-resolution display (1600x1200) running at 32-bit color depth requires about 7.3MB of video memory. Today's video cards invariably have at least 8MB, and most ship with 32MB, 128MB, or more. You might run into the RAM problem if you're using a very old video card, though—say, one made before 2000 or so. If this is the case, you may need to reduce the bit depth in order to run at your chosen resolution or reduce the resolution to run at the chosen bit depth. If this is a problem, you should consider upgrading your video card. Even a sub-\$20 modern card will probably perform better than an antiquated one. Be sure you get a card that works with your motherboard, though; many modern cards require recent Accelerated Graphics Port (AGP) slots, which older motherboards lack.

The second problem (time) is more of an issue. If you want the zippiest display possible, you should run at the lowest color depth you can tolerate. Just how low you should go in this limbo, though, depends on your personal preferences and the capabilities of your hardware. A modern video card should be able to deliver performance that's adequate for most people even at 32-bit color depth. A 24-bit color depth is adequate for almost all tasks. The human eye has a hard time distinguishing different colors beyond the 16 million delivered at that depth. Even a 15- or 16-bit color depth is adequate for many purposes; Linux desktops continue to look good at these resolutions. Dropping below this level can cause problems, though. At 8-bit color depth, Linux will have to reduce the number of colors in many programs' icons, resulting in a somewhat ugly look to many programs. Digitized photos will also look bad at 8-bit color depth. The 2-bit color depth, although supported by X, is intended mainly for use on certain very old video cards, which work only in this two-color mode. (This mode resembles very old monochrome Macintosh displays.)

Changing X Color Depth Options

Actually setting the X color depth requires editing your X configuration file, `xorg.conf`, `XF86Config`, or `XF86Config-4`, located in `/etc/X11` or `/etc`. Locate this file and look for lines that resemble the following:

```

Section "Screen"
    Identifier "screen1"
    Device "device1"
    Monitor "monitor1"
    DefaultDepth 24

    Subsection "Display"
        Depth 24
        Modes "1280x1024" "1024x768" "800x600"
    EndSubsection
EndSection

```

This Screen section defines both the resolution and the color depth of the display. It includes one or more Display subsections, which associate a set of resolutions with a given color depth. For instance, this example tells X that it should try to use the 1280x1024, 1024x768, and 800x600 resolutions, in that order, when told to use a 24-bit color depth. When X starts, it tries to use the color depth specified by the DefaultDepth line; it looks up the appropriate Display subsection for that depth and tries to use each of the resolutions it defines.

To change the default color depth, you should change the DefaultDepth line to list the depth you want to use. You must also ensure that an appropriate Display subsection exists for that color depth. If one doesn't exist, copy an existing entry (the previous default should be a good choice) and change the Depth line of the copy.

When you're done, save your changes and log out of your X session. On most distributions, this restarts the X server. Some distributions provide an X restart option at the GUI login screens, so you can try that, too. If you can't find one and you don't believe X has restarted, press **Ctrl+Alt+Backspace**; this shuts down X, which should then restart. As a last resort, try rebooting the computer. When X starts up again, it should be running in the new color depth.

Setting the Resolution

You can change the display resolution just as you can change the color depth, by editing the Screen section in the X configuration file, as just described. The difference is, instead of editing the DefaultDepth line, you edit the Modes line to start with your preferred video mode and to list all your other acceptable

video modes. For instance, suppose your desktop is too small for your taste; it's set to 800x600, but you want to use 1280x1024. Chances are you'll see a line like this in the `Display` subsection for your chosen color depth:

```
Modes "800x600" "640x480"
```

NOTE: If you're certain that your current color depth setting lists your preferred resolution before the resolution you're using, chances are either your monitor can't handle the higher resolution or you've set your monitor's refresh rate incorrectly. Correcting the latter problem is described next, in "Setting the Refresh Rate."

To change the configuration to support a higher resolution, add or change your desired resolution, but be sure it comes *before* any other resolutions on the display list:

```
Modes "1280x1024" "800x600" "640x480"
```

Including lower resolutions on this list isn't harmful, and in fact it can be helpful. Simultaneously pressing the Ctrl key, the Alt key, and the + or – keys on the numeric keypad cycles through the various resolutions. In lower resolutions, X presents a *virtual desktop*—a lower-resolution “window” on a larger desktop. You can move around it by moving the mouse cursor to the edge of the physical screen, whereupon the screen scrolls. This feature can be handy to get a quick close-up look at something that's on the screen.

GunkBuster's Notebook: Setting the Resolution for LCDs

In 2004, liquid crystal display (LCD) monitors began to outsell traditional cathode-ray tube (CRT) monitors. LCDs have also long been the dominant display type on laptop computers. LCDs are smaller, lighter, more energy efficient, and more expensive than their CRT counterparts. LCDs do have an unusual drawback: They have a fixed number of pixels. For instance, most 17-inch LCD monitors have 1,280 pixels across and 1,024 pixels down. This feature means that they can handle 1280x1024 displays quite well, but when the video card outputs most other resolutions, such as 1024x768, the monitor will either display a smaller image or create an ugly display by interpolating values.

For best results, you should ensure that X is running in the appropriate resolution for your LCD monitor. Consult your monitor's manual to determine what this resolution is, if you don't know.

Setting the Refresh Rate

Computer monitors change, or refresh, their displays a fixed number of times per second. This *refresh rate* is expressed in Hertz (Hz), as in a 75Hz refresh rate. As a general rule, the higher the refresh rate, the better; the human eye perceives low refresh rates as flicker, and even below consciously perceptible levels, flicker can cause headaches and eye strain.

A CRT monitor's overall refresh rate is determined by two separate values, often referred to as the *horizontal refresh rate* and the *vertical refresh rate*, or minor variants of these terms. Modern monitors support limited ranges of horizontal and vertical refresh rates, expressed in kHz and Hz, respectively. X can compute the optimum overall refresh rate based on the monitor's capabilities and your selected video mode, but to do so, it needs to know the monitor's capabilities. To tell X about these capabilities, you must locate the Monitor section of your X configuration file:

```
Section "Monitor"
    Identifier "monitor1"
    VendorName "LiquidVideo"
    ModelName "L17LCD2"
    HorizSync 30-80
    VertRefresh 55-75
EndSection
```

Your Monitor section may include additional lines that set more features. If so, ignore them. The important lines to adjust are the HorizSync and VertRefresh lines, which set the horizontal and vertical refresh rates, respectively. If your CRT monitor is flickering, you should verify that these values are set correctly. To do so, though, you *must* have documentation on your monitor. Monitors' manuals usually provide this information in a specifications section. If you don't have your monitor's manual, try searching on the manufacturer's Web site or doing a general Web search.

WARNING! Do not enter random guesses for the HorizSync and VertRefresh lines! Older monitors can actually be damaged by out-of-spec signals, so guessing at these values has the potential to destroy your monitor! Modern monitors are usually smart enough to ignore such signals, but it's best to play it safe on this score.

Once you've made these changes, restart X, as described earlier, in "Changing X Color Depth Options." The result should be that X will begin using a higher refresh rate, if your changes enable it to do so.

NOTE: LCD monitors that accept analog inputs (that is, inputs designed for CRT monitors) also have horizontal and vertical refresh rate settings, so you can adjust them in the same way as described here. LCD monitors, though, aren't affected by these settings in the same way as CRT monitors. Thus, if your LCD monitor works in the resolution you're using, adjusting the refresh rate isn't likely to visibly change the monitor's flicker, as with a CRT display. You might see some subtle changes in the stability of individual pixels, though.

Enabling 3D Optimizations

X was designed as a two-dimensional (2D) display system; it displays text, dialog boxes, and so on in a basically 2D way on a 2D monitor. Although windows can be displayed “in front of” other windows, this effect isn't really a full three-dimensional (3D) effect.

A few programs, though, make use of 3D display features. The most common example is certain games, particularly those that present a first-person perspective on a 3D game environment. Of course, the display itself isn't truly 3D, but the game creates an illusion of a 3D world, much as a photograph or movie presents a 2D projection of a 3D world.

Unfortunately, the computations involved in creating a realistic 3D display are quite intense. Even the fastest of CPUs can be brought to its knees trying to perform all the subtle computations necessary to create a convincing 3D world. Thus, video card manufacturers have developed 3D video cards. These cards include graphics processing units (GPUs) that are designed specifically to perform the specialized computations needed to create a 2D projection of a 3D virtual world. The result can be greatly improved performance and graphics image quality in first-person games and other applications that use 3D effects. The problem with this hardware is that it requires special software, and that software isn't available for all video cards under Linux. Cards that lack 3D acceleration supports (which are extremely rare today) or for which Linux drivers aren't available can still run 3D programs, but the performance of these programs will be quite poor compared to the same programs run on video cards with full Linux 3D support.

The main Linux 3D support relies on the OpenGL specification, which is a platform-independent 3D application programming interface (API). In theory, a program written to use OpenGL can be easily ported to other platforms that support OpenGL, such as Windows. OpenGL is an API specification, not a program. Thus, to use OpenGL, you need software that implements it. This software ships as part of XFree86 4.x and X.org-X11, but it may not be loaded by default. At a minimum, you should check the `Module` section of your X configuration file and verify that it contains the following line:

Load "glx"

This line loads the GLX module, which is X's main OpenGL software package. Unfortunately, matters get complicated from here on because each video card requires its own OpenGL support. As I write, this support is available from several sources:

- ✓ **DRI**—This package is built into XFree86 4.x and X.org-X11. It includes hardware acceleration support for a wide range of video chipsets, but by no means for all of them.
- ✓ **Utah GLX**—This open source project, located at <http://utah-glx.sourceforge.net>, delivers hardware drivers for several nVidia, ATI, and Matrox chipsets, with "bleeding-edge" support for some S3 cards. Not all chipsets from all manufacturers are supported, though.
- ✓ **nVidia**—This hardware manufacturer provides its own X drivers with OpenGL support. This driver tends to be well liked by nVidia users, although some users avoid it because it's not open source. Check the nVidia Web page, <http://www.nvidia.com>, for details.

If you're lucky, DRI will support your card. To use it, you must include a Load "dri" line in your X configuration file's `Module` section. This module in turn uses its own DRI section:

Section "DRI"

```
Group      0
Mode      0666
```

EndSection

The main point of this section is to set permissions on various DRI-specific system resources. This example does so with the `Mode` line, which specifies a permission mode similar to that used on ordinary files. In this case, 0666 is used, which gives all users full access to the accelerated 3D hardware.

Once you've configured X to use your 3D hardware, restart X (as described earlier, in "Changing X Color Depth Options") and test the system by typing **glxgears**. This runs a simple OpenGL application, as shown in Figure 12-1; the three gears in this window spin constantly, and every few seconds, a report of the rendering engine's speed, in frames per second (fps), appears in the console you used to launch the program. You should leave the `glxgears` window on the screen and unobscured by other windows for several seconds to see what sort of scores you're getting. Even a very fast computer without working



Figure 12-1

The `glxgears` program is a simple OpenGL application that can be used to assess the speed of your OpenGL subsystem.

hardware 3D drivers isn't likely to exceed 100 fps. If you have working 3D acceleration, you should see fps scores of several hundred to several thousand.

If you believe your 3D acceleration should be working but you get `glxgears` scores that are disappointing, you should review your configuration. Be sure the `glx` and (if you're using DRI) `dri` modules are being loaded—but non-DRI drivers require that the `dri` module *not* be loaded. Review the documentation for your particular video card's drivers, because each one may require its own quirky configuration.

Configure X Fonts

In the past, a common complaint concerning Linux and GUI applications has been that fonts look bad. Although this complaint is heard less often today than it was a few years ago, X's font system is still rather confusing, and font problems can still rear their gunky heads. Fortunately, you can clean up the mess with the help of some understanding and a bit of configuration elbow grease. To begin, you must understand how Linux handles fonts. Of course, you may also need to locate some fonts worthy of use, so I describe that task briefly. With basic knowledge and fonts in hand, you can configure Linux's two main font

systems, *X core fonts* and *Xft fonts*. Finally, I provide some tips on how to go about managing your fonts to keep them from gunking up your system.

Understanding Linux Fonts

Linux's font handling can be confusing because Linux relies on several font-handling subsystems:

- ✓ **X core fonts**—The most fundamental Linux font system is the X core fonts system, which is the font system that X directly supports. X core font support is built into X, and it offers certain advantages, such as the ability to use fonts delivered by a special network font server. On the other hand, X core fonts are rather limiting. They don't support certain modern features, such as *font smoothing* (aka *anti-aliasing*), in which curves and diagonal lines use some pixels with shades of gray rather than black or white in order to reduce the jagged appearance of these features. X core fonts also don't provide any easy way to link up with printed fonts, which complicates the design of word processors and other print-centric programs.
- ✓ **Xft fonts**—In an effort to work around the problems of X core fonts, a new system, known as Xft fonts, has been developed. Xft fonts support font smoothing and various features that are helpful to software developers. Most modern KDE and GNOME programs, among others, use Xft fonts in preference to X core fonts.
- ✓ **Application-specific fonts**—Some programs (mostly word processors and other programs that need to do sophisticated things when printing) support fonts directly, bypassing X core fonts. Most such programs adopted this approach prior to the development of Xft fonts, and most such programs that are still being maintained are transitioning to Xft fonts.

Because of this mishmash of font systems, in order to install a font in Linux, you may need to install it several times—once as an X core font, once as an Xft font, and perhaps in individual font-using programs such as word processors. (Chapter 5 describes installing fonts in OpenOffice.org, for instance.)

Obtaining Good Fonts

Most Linux distributions ship with a wide variety of fonts, but the details vary greatly from one distribution to another. Two key font sets are almost always present, though: clones of several common fonts found on PostScript printers, provided as part of Ghostscript, and the Bitstream Vera fonts. These fonts are all useful, but you may want to move beyond them. You have several options, including the following, for doing so:

- ✓ **Microsoft's core Web fonts**—Several years ago, Microsoft released several fonts for use on the Web and made them freely available for download and use, even on non-Microsoft OSs. Although Microsoft has terminated this program, the license terms of the fonts mean that others may still distribute them for non-commercial purposes. Check <http://corefonts.sourceforge.net> for information on how to obtain and use them. Some distributions provide an easier way to install these fonts; for instance, you can use Debian's `apt-get` to install the `msttcorefonts` package or Gentoo's `emerge` to install the `corefonts` package.
- ✓ **Commercial font collections**—Walk into any computer superstore and you'll find CD-ROMs filled with fonts. Most of these font collections feature fonts of questionable utility, but most also have at least a few fonts that are reasonably good. More expensive font collections are available from companies such as Bitstream (<http://www.bitstream.com/fonts>), Adobe (<http://www.adobe.com>), and LinoType (<http://www.linotype.com>), both on CD-ROMs and on the Web. These fonts typically look better on the screen and when printed than fonts from the cut-rate commercial CD-ROMs.
- ✓ **Fonts in commercial software**—Some commercial software products, such as word processors and graphics packages, ship with their own fonts. The Corel (<http://www.corel.com>) Draw package is noteworthy as coming with a wide selection of fonts, and old versions of Corel Draw and other font-containing programs can often be found in closeout bins and on eBay for very low prices. Thus, you might consider buying such a program just for the fonts.
- ✓ **Low-cost and free fonts on the Internet**—Many Web sites are devoted to distributing free or low-cost fonts. Examples include <http://www.1001freefonts.com>, <http://www.fontfreak.com>, and <http://www.free-fonts.com>. One site that's particularly interesting from a Linux perspective is the Comprehensive TeX Archive Network (CTAN) font collection, <http://www.ctan.org/tex-archive/fonts/>. This site holds fonts that have become favorites of users of the TeX layout software, which is popular on Linux.

TIP: *If your system dual-boots between Linux and Windows, you can mount the Windows partition in Linux and use its font directory directly. This will give you access to Windows fonts in Linux with minimal fuss. If you use the New Technology File System (NTFS) under Windows, though, you won't be able to write Linux font description files to the font directory from Linux. You can either copy these files from Linux to Windows manually (but creating the files will be awkward) or copy the entire Windows font directory to a native Linux directory and use them from there.*

If you've seen a font in print that you like, finding a lookalike font can be difficult. One site that can help is WhatTheFont (<http://www.myfonts.com/WhatTheFont/>). This site enables you to upload a graphic file of a font, such as a scan of a few words in a magazine. The site then tries to match the font to its database, pointing you to the font the source's designers used, or at least to something that looks similar.

When you look for fonts, try to get Type 1 or TrueType fonts in PC (Windows) format. Mac format fonts and fonts in more exotic formats must be converted first, and that process isn't always easy. If you have a choice, I recommend using Type 1 fonts, simply because they often produce cleaner Portable Document Format (PDF) documents if you need to create a PDF file using the font.

TIP: If you need to convert a font from Mac format or between font types (TrueType to Type 1, say), check out FontForge (<http://fontforge.sourceforge.net>). This program is a Linux font editor that includes the ability to read fonts in a wide variety of formats and save them in an equally wide range of formats. It's overkill for mere format conversion, but it's also the best program I know of for doing this job.

Type 1 fonts use a variety of filename extensions. Most commonly, .pfa and .pfb denote the main font file (only one is necessary), .afm and .afb denote a "helper" file with extra font spacing information, and files with various other extensions can provide additional information. Only the main font file (with a .pfa or .pfb extension) is necessary to install the font, but some programs that deal with fonts more directly may require additional files. TrueType fonts are simpler in this respect; they consist of a single file with a .ttf filename extension.

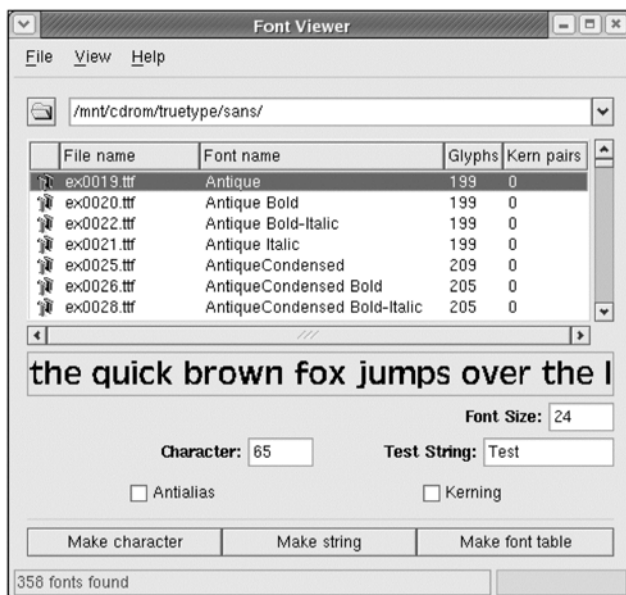
No matter what font format or source you use, you may want to preview your fonts before installing them. This can be done with tools like gfontview (<http://gfontview.sourceforge.net>), which enables you to browse through a directory and see a short bit of text in any font file you select, as shown in Figure 12-2. FontForge, mentioned earlier, can also serve this function.

Configuring X Core Fonts

X core fonts are the most tedious of the font types to configure. The procedure involves first preparing a font directory that's to hold the fonts and a file describing them, followed by telling X about this font directory.

Preparing Font Directories

An X font directory can be any directory on the computer. In practice, X.org-X11 uses subdirectories of /usr/share/fonts and XFree86 uses subdirectories

**Figure 12-2**

Previewing fonts before installing them can help prevent the accumulation of font gunk.

of `/usr/X11R6/lib/X11/fonts` for their default fonts. If you're adding new fonts, you can place them in any directory you like. I recommend you create a subdirectory of `/usr/local` or `/opt`, as in `/opt/fonts`. This will keep your own fonts distinct from the standard system fonts. You might even want to create multiple subdirectories for fonts from different sources or in different formats, as in `/opt/fonts/type1` and `/opt/fonts/tt` for Type 1 and TrueType fonts, respectively.

Once you've placed font files in a font directory, you need to prepare a description of the fonts in this directory. This description appears in a file called `fonts.dir`. Although you can theoretically create this file by hand, its format is sensitive and a bit tedious. Thus, you should use a configuration tool to help with this task. If you're using X.org-X11 or XFree86 4.3 or later, you can do the job with `mkfontscale` and `mkfontdir`. The first program creates an intermediate file, `fonts.scale`, based on the contents of the current directory, while `mkfontdir` converts `fonts.scale` into a `fonts.dir` file. To use these programs, simply type their names in the font directory:

```
# mkfontscale
# mkfontdir
```

The result should be a usable `fonts.dir` file. You can peruse this file if you like, but don't try to edit it by hand unless you know about its format. The main features you're likely to notice are the font filenames and the font names extracted from the files themselves.

Telling X about Your Font Directories

To actually use any fonts you've installed, or to remove entire font directories from an existing configuration, you must normally edit the X configuration file, `xorg.conf`, `XF86Config`, or `XF86Config-4`. Whatever its name, this file holds a `Files` section with a series of `FontPath` entries:

```
Section "Files"
    FontPath "/usr/share/fonts/local/"
    FontPath "/usr/share/fonts/misc/"
    FontPath "/usr/share/fonts/100dpi:unscaled"
    FontPath "/usr/share/fonts/Type1/"
    FontPath "/usr/share/fonts/TTF/"
    FontPath "/usr/share/fonts/100dpi/"
EndSection
```

NOTE: If you don't see entries like this but you do see a `FontPath` entry that refers to `"unix/:-1"` or `"unix/:7100,"` your system is using a font server for font delivery. This configuration is most common on Fedora, Red Hat, and Mandrake systems. You can still add new font directories directly to X with these systems; you'll just have to add them either before or after the single existing `FontPath` entry that points to the font server.

To add your prepared font directory to X, copy one of the existing `FontPath` lines and modify it to point to your new font directory. For instance, if you've added fonts to `/opt/fonts/type1` and `/opt/fonts/tt`, you would add the following lines to the list:

```
FontPath "/opt/fonts/type1/"
FontPath "/opt/fonts/tt/"
```

The location of the new entries in the list determines the search order for fonts. This can influence the speed with which fonts appear on the screen, and it also affects which font takes precedence if two directories contain fonts with the same name—the directory that comes first takes precedence.

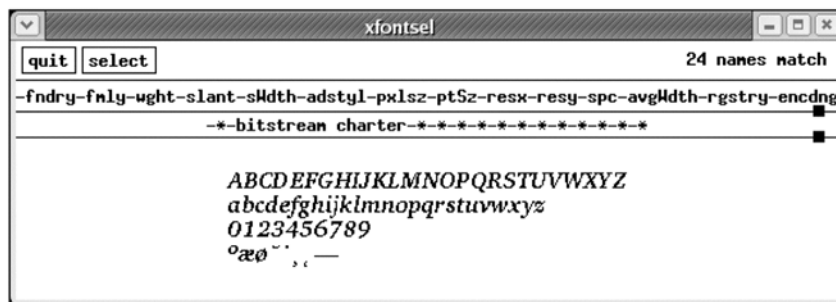


Figure 12-3

The xfontsel program displays installed X core fonts.

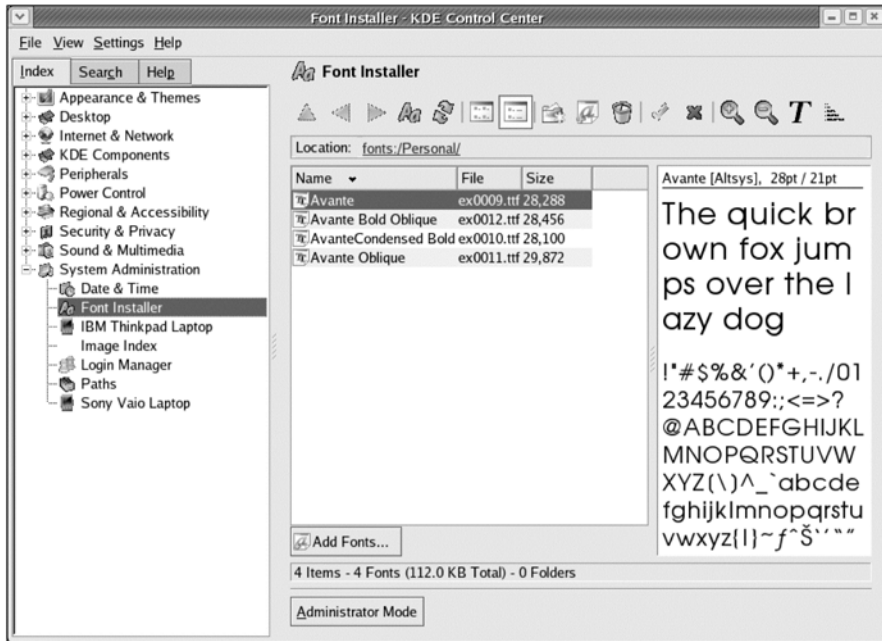
Once you’ve made these changes, save the file and restart X, as described earlier, in “Changing X Color Depth Options.” You should now be able to use your new fonts in applications that use X core fonts. To verify that the fonts are installed, type `xfontsel`. This command brings up a basic font display tool, as shown in Figure 12-3. Click in the Fmly menu to see a list of font families installed. You should see your new fonts there, and when you select one of them, it should display in the main window.

Many modern programs don’t use X core fonts, so you might not see your fonts in these applications even after installing your fonts as X core fonts. If this happens, you should read the upcoming section, “Configuring Xft Fonts.” Some word processors have their own font-installation routines, as well.

Configuring Xft Fonts

On the whole, Xft fonts are easier to configure than are X core fonts. In fact, some desktop environments provide tools to help with this process. For instance, Figure 12-4 shows KDE’s Control Center open on its Font Installer, which is accessible from the System Administration area. (Open the Control Center from the KDE menu or type `kcontrol` in a shell prompt window.) To install fonts for personal use, click Add Fonts. The result is a dialog box in which you select a font file, which could reside anywhere on your system, including on a font CD-ROM. Select one or more files and the tool copies them to your `~/ .fonts` directory and makes them accessible as Xft fonts. You may need to restart open applications to use the fonts, but you don’t need to restart X.

By default, the KDE Control Center’s Font Installer installs fonts in your home directory. You can also use it to install fonts system-wide, though. To do so, click Administrator Mode. You must then enter the root password, whereupon you can use the Control Center to add fonts to any existing system font directory.

**Figure 12-4**

KDE's Control Center provides a Font Installer utility you can use to install fonts.

One problem with using KDE Control Center to add fonts is that it only reports font directories in the directories that it knows about. If you've prepared one or more new font directories, as described earlier in "Preparing Font Directories," you can add them to your Xft configuration by editing the `/etc/fonts/local.conf` file. This file contains local font configuration information—that is, configurations that you change yourself. Load this file into your favorite text editor as root and look for any font directory lines. These lines are delimited by `<dir>` and `</dir>` strings:

```
<dir>/usr/local/share/fonts</dir>
```

NOTE: This configuration applies to Xft 2.0, which ships with X.org-X11 and recent versions of XFree86. Most early versions of XFree86 4.x shipped with Xft 1.0 or 1.1, which used a different configuration file. XFree86 4.0.1 and earlier didn't support Xft at all. If you're using a very old version of X, you won't be able to use Xft fonts.

If such lines are present, add lines just like the existing ones for the font directory you've created. Xft looks in the directories you specify and all their subdirectories, so if you've created, say, an `/opt/fonts` directory that holds two or three subdirectories, you need only create an entry for `/opt/fonts`, not for all of its subdirectories.

If your `/etc/fonts/local.conf` file lacks any font directory listings, add them, complete with `<dir>` and `</dir>` codes around each line you add. Make your additions just *before* the `</fontconfig>` line; this line marks the end of the file, so you shouldn't add anything after it. You should also keep your entries out of blocks of lines that are delimited by `<!--` and `-->` strings; these denote the start and end of comment lines, and any lines between these strings are ignored.

Once you've made these changes, type **fc-cache** as root. This command causes Xft to go through all the current font directories and build `fonts.cache-1` files for each directory. These files contain font information that helps Xft present font lists. If you forget to do this, fonts will still work, but Xft will place the equivalent information in each user's home directory. This will slow down the startup of Xft-using applications the first time they're launched after adding the fonts.

Managing Fonts

One of the challenges of font management is keeping them from gunking up your system. Everybody who uses a word processor likes playing with fonts, but sooner or later, the novelty wears off, and those hundreds of fonts get to be a drag, as they can slow down your use of the system. Locating the handful of fonts you use in a long list can be a problem, and programs may respond more slowly to certain actions as they have to wade through many fonts when presenting certain menus or dialog boxes.

The key, really, is to exercise moderation. Don't add fonts unless you're reasonably sure you'll actually use them, and go through your additions every now and then to be sure they're fonts you actually do use. Remember to clean out both the fonts you've added globally (by editing `/etc/fonts/local.conf` or using a system administrator option in a GUI font management tool) and the fonts you've added to your account (via a GUI font management tool's local options or by adding the fonts directly to `~/.fonts`).

One complication to font management is that some distributions ship with a wide selection of free fonts, many of which will inevitably be gunk for any individual user. The trouble is that you can't simply delete all your distribution's font packages because some really are necessary. You can begin to get a handle on the situation by tracking down the font packages. Use your package management tools, as described in Chapter 6, "Cleaning Out Unused Packages," to identify the packages to which individual font files and directories belong. For instance, suppose you've identified the fonts in `/usr/share/fonts/extraTT` as fonts you don't need. If your system uses RPM, you could type the following command to identify the packages that own the files in this directory:

```
$ rpm -qf /usr/share/fonts/extraTT/*
```

The result will be a long series of output lines identifying font package names. You can then use RPM to look up the files that belong to the packages:

```
$ rpm -ql extraTT
```

The result is a long list of files contained in the package. If this list doesn't contain any fonts that you know you need, you can try removing it:

```
# rpm -e extraTT
```

You should keep a couple of caveats in mind when removing fonts:

- ✓ When you remove a font package, programs may become confused. In the case of programs that use X core fonts, this can happen even when launching programs long after removing the fonts. It's therefore best to restart X after removing fonts or remove fonts when X is shut down.
- ✓ X relies on the presence of certain core fonts and may not work correctly if they're missing. As a general rule, these fonts live in the 75dpi, 100dpi, misc, Type1, and Speedo subdirectories of your main font directory (/usr/share/fonts or /usr/X11R6/lib/X11/fonts). Thus, you should be very wary of removing fonts from these subdirectories; however, some font packages add fonts to these directories (particularly the Type1 directory). In the end, you may need to use some common sense. A Klingon font isn't likely to be critical, even if it's in a core font directory, but a common font like Times might be.

Summing Up: Keeping Your Display Clean

X is a tricky Linux component to configure, but it's unusually important for desktop use of the OS. Picking the right driver and configuring the system to use a good color depth, resolution, and refresh rate will go a long way toward keeping your display reasonably speedy. Although it's not really critical for most desktop applications, 3D acceleration can help with games and a few more exotic tools, but configuring hardware 3D support requires digging into very hardware-specific drivers and their documentation. No matter what hardware you're using, fonts are important for maintaining a legible display. X's font handling is complex and involves two main font systems. Thus, installing fonts in X can be a bit involved, but installing good fonts and keeping your system free of extraneous fonts can help keep your system looking good.

Appendix A



Backing Up Linux

You can clean up most types of gunk by applying common sense, know-how, and popular Linux tools. Sometimes, though, gunk can grow out of control like kudzu or a B movie monster. In the worst cases, you might have no choice but to start your Linux installation over again. For instance, a gunky hard disk could die and take all your data with you, or a security breach might leave you unable to trust any of the system software on the computer. In such situations, a backup is invaluable. If you have a backup from before the problem developed, you can completely wipe out the trouble spot and restore the backup, effectively turning the clock back and making your computer what it was before anything went wrong. In some cases, you might then need to take steps like updating your software or upgrading your security to ensure that the problem doesn't recur, but having a backup can save you a lot of configuration hassle. Backups of your user data files can save hundreds of hours of work.

To back up your computer, you must prepare by selecting appropriate backup hardware and software. Actually performing the backup is usually the easy part, but you must do it on a regular basis and keep track of your media, lest you discover that your latest backup is months old. Finally, no backup will do you any good unless you can restore it, so you should have a recovery plan.

Selecting Backup Hardware

When designing a backup system, you should begin with your choice of backup medium. Several media are popular for backups:

- ✓ **Tapes**—Magnetic tape has long been the traditional backup medium. Tape drives range in price from a couple hundred dollars to several thousand, with individual tapes costing a few dollars to tens of dollars. Magnetic tape tends to be a bit slow and unreliable, but details vary with the type of tape—several varieties exist. Overall cost tends to be low, particularly for high-volume backup needs, such as network backup servers. (For such systems, the cost is high in absolute terms but low on a per-gigabyte basis.) Tape's cost advantage is being eroded by the dropping prices of conventional hard disks, though.
- ✓ **Optical discs**—Optical discs (CD-Rs, CD-RWs, and various recordable DVD technologies) are good choices for relatively small backups, such as user data backups on modest systems. Readers for these discs are ubiquitous, which makes data recovery easy. Most estimates place the anticipated lifetime of optical media in the decades, which makes them good for archival storage. (Some recent studies are more pessimistic, though. For best longevity, *do not* use stick-on labels; leave discs unlabeled, use a disc printer to print a label directly on the disc, or mark them with permanent felt-tip markers.) Optical discs are inexpensive on a per-gigabyte basis.
- ✓ **Removable disks**—Removable disk technologies range from floppy disks to high-capacity media such as Orb and Jaz disks. These are convenient storage media for individual projects and perhaps user backups, but they tend to be expensive on a per-gigabyte basis, and most drives have low capacities. This is particularly true of floppy disks, which are also extremely slow.
- ✓ **Removable hard disks**—You can purchase a special mounting bay and trays for hard disks to turn them into high-capacity removable disks. The dropping price of hard disks makes this approach cost competitive with many tape formats and removable hard disks are more convenient in many ways. Hard disks are relatively fragile, though, so you must take care in storing the disks. This feature complicates off-site storage, as well. Overall, this format can be a good choice for full system backups or user backups when users store very large files.

Overall, tapes and removable hard disks are the media of choice for backing up your entire system. For backing up smaller subsets of data, optical discs and smaller removable disks can be good choices.

Fortunately, all of these backup media are Linux compatible, provided they're used with Linux-compatible interfaces. ATA, SCSI, and USB interfaces are all popular, so the main issue is ensuring that Linux drivers for the interface hardware exist.

NOTE: Some SCSI tape drives come with very inexpensive SCSI host adapters, which might not have Linux drivers. If you buy a SCSI tape drive and can't get the host adapter to work, you may need to replace it with a Linux-compatible model.

Selecting Backup Software

Your choice of backup software will depend in part on your choice of backup hardware. The best software to use for one type of hardware might not be optimal (or might not work at all) for another type of hardware. Popular options include those listed here:

- ✓ **tar**—The Linux tar program is frequently used for creating on-disk archive files, but it's also a popular backup tool. You can write directly to tape drives using tar, and you can create archives on removable disks, including removable hard disks. You can use tar in conjunction with optical media, but you'll need extra software to do the job.
- ✓ **cpio**—This program is an archiving tool similar to tar in capabilities. Some people prefer cpio to tar, but tar is more popular.
- ✓ **dump**—The dump program is a traditional Unix backup program that's also available for Linux. Unfortunately, it's very filesystem specific, and as I write, only versions for ext2fs, ext3fs, and XFS are available. You can't use dump to back up ReiserFS or JFS partitions, much less non-Linux filesystems. Despite the fact that it's referred to in a lot of Unix documentation and even some Linux system files, dump isn't the best choice for backups.
- ✓ **cp**—The Linux cp command, which copies files, can be used to back up to removable disks, including removable hard disks. Mount a disk at some location, such as /mnt/backup, and use the -a option to cp to copy files to that location, as in **cp -a ./mnt/backup** to copy the current directory and all its subdirectories to the backup disk.
- ✓ **Optical disc tools**—Optical discs present unique challenges. Under Linux, the cdrecord program writes to an optical disc, but it requires a prepared file. One option is to create a tar archive and write it directly to disc using cdrecord. More commonly, though, you'll either use mkisofs to create an ISO-9660 filesystem of the files you want to back up or use mkisofs to store a tarball of the files in an ISO-9660 filesystem. You'll then write this ISO-9660 filesystem to disc with cdrecord. GUI front ends to

these tools, such as X-CD-Roast (<http://www.xcdroast.org>), ERoaster (<http://eclipt.uni-klu.ac.at/eroaster.php>), and K3b (<http://www.k3b.org>), can help simplify the process.

- ✓ **Commercial and high-end tools**—Large installations frequently make use of more sophisticated tools, some of which are commercial. These programs simplify the backup of an entire network of computers.

Because of its popularity (including its use in non-backup operations) and the fact that it's something of a “lowest-common-denominator” backup tool, I describe tar in more detail shortly. You can apply these principles if you choose another tool, though. In fact, if you want to back up to optical discs, you *must* use other tools instead of or in addition to tar.

Performing a Backup

The tar program accepts a large number of functions and options. You specify one function and may optionally specify an arbitrary number of options. The functions tell tar what type of action it's to perform, such as create a tar file, extract data from a tar file, or list a tar file's contents. The options modify how it's to perform these actions, such as telling tar whether or not to call an external compression program or display a listing of files as it's backing them up.

A typical tar backup command looks like this:

```
# tar --create --verbose --one-file-system --file /dev/st0 /home /usr
```

This command can be abbreviated as follows:

```
# tar cvlf /dev/st0 /home /usr
```

These options tell tar to do certain things:

- ✓ **Create a backup**—As you might expect, the `--create` command (abbreviated `c`) tells tar to create a backup. Other commands include `--extract` (`x`) to restore files and `--list` (`t`) to list files in an existing backup.
- ✓ **Be verbose**—The `--verbose` (`v`) option tells tar to display each filename as it's backed up.
- ✓ **Back up a single filesystem**—The `--one-file-system` (`l`) option results in a backup of just one filesystem. For instance, if your system has root (`/`), `/home`, and `/usr` partitions, backing up the root filesystem without this option backs up all three partitions, but doing the same with this option backs up just the root partition, not the `/home` or `/usr` partitions. This

option should always be used in backups because Linux employs some virtual filesystems, such as `/proc`, which should not be backed up, as they would be if you backed up the root filesystem without this option.

- ✓ **Specify a target file**—The `--file (f)` option specifies a filename, which immediately follows, in which the backup is to be stored. In this example, this file is `/dev/st0`, which corresponds to a SCSI tape device. If you have an ATA tape drive, you'd use `/dev/ht0` rather than `/dev/st0`. If you're backing up to a removable disk, you'd mount it and specify a file on the mount point.
- ✓ **Specify directories to be backed up**—The final options are the files or directories to be backed up—`/home`, `/`, and `/usr` in this example. The order of these items means that files stored in `/home` will be the quickest to restore, followed by those in the root partition, followed by those in `/usr`.

Although I didn't show it in this example, another option you might want to use is `--zip (z)`, which compresses data. The `--bzip2 (j)` option does an even better job of compressing data, but it takes more CPU time. Most mid-range and high-end tape drives provide compression themselves, so this option is often unnecessary when using tapes. In fact, the tape drive's built-in compression is less likely to cause problems if there's a read error when restoring data, so using the tape drive's compression is preferable. You might want to compress data when backing up to removable disks or preparing a tar file for subsequent storage on an optical disc, though.

If your backup medium isn't big enough to store a whole backup, you have two choices for how to proceed:

- ✓ Back up your system in chunks by specifying enough directories to fill one medium, then another, and so on.
- ✓ Use the `--multi-volume (M)` option, possibly in conjunction with the `--tape-length N (L)` option, to have tar stop backing up when the medium fills up (or when the specified number of kilobytes have been backed up).

You should always perform full system backups as root, because only root has access rights to all of the files on the computer. Ordinary users can usually back up their own files, although ordinary users often lack write permission to the tape device, so you might need to change those permissions if you really want to enable users to write directly to tape.

WARNING! Full system backups contain extremely sensitive files, such as password files. Thus, you should protect your backup media very carefully, lest an unscrupulous individual acquire them and use them to break into your system.

Restoring Data

Having a backup won't do you much good if you can't restore data. In the best-case scenario, you need only restore a few files. Sometimes, though, you may need to be ready for a fuller disaster recovery restore.

Restoring a Few Files

To restore just a few files on a system that works, you can run the tar operation in reverse. For instance, suppose you've accidentally deleted the important.sxw file from your home directory (/home/mydir). The following commands will restore the file:

```
# cd /
# tar xvf /dev/st0 home/mydir/important.sxw
```

This procedure is the opposite of the original backup command, but with a few twists:

- ✓ The first command changes to the root (/) directory, because all backups are stored relative to that directory. If you don't do this, the backup file will be restored to the home/mydir subdirectory of the current directory.
- ✓ The file specification is for just the one file, and it omits the leading slash (/) on the filename. This is because filenames are stored in tar archives without this slash, so including the slash would result in tar failing to find the file in the backup.
- ✓ The `-one-file-system (1)` option is omitted, because it's normally unnecessary on the restore side.

TIP: If you're not sure of the exact filename of the file you want to restore, use the `-list (t)` command to obtain a listing of all the files in the backup. You might redirect the output to a file, as in `tar tvf /dev/st0 > files.txt`. You can then peruse files.txt to find the name of the file you want to restore.

If you want to restore an entire directory, you can do so; simply specify a directory name rather than a filename. Similarly, you can restore multiple files or directories at once by listing all of their names, just as you can do when backing up.

Performing a Full Emergency Restore

Restoring a few files is fairly straightforward. The real trick comes when you encounter the La Brea Tar Pits of gunk—when a hard disk crashes or some other disaster strikes that wipes out your entire computer. In this situation, you must restore your backup without the benefit of your normal working Linux system. The best solution is usually to use a Linux emergency boot CD-ROM, such as Knoppix (<http://www.knoppix.org>). These systems are bootable Linux installations on CD-ROM. You stick one in your CD-ROM drive and it boots, enabling you to access your backup and restore it using normal Linux commands. In a worst-case scenario, you might need to perform all of these steps:

1. Replace any broken hard disk.
2. Boot your emergency Linux system.
3. Create new partitions for your Linux system on the new disk. Linux's `fdisk` program is the traditional tool for this job, but GNU Parted (via the `parted` command) is another option. Full GUI emergency systems may also support QTParted, which is a GUI front end to Parted.
4. Create filesystems on the new partitions. GNU Parted and QTParted can do this as part of their operation, but if you use `fdisk` or don't explicitly create partitions in GNU Parted or QTParted, you must create filesystems with `mkfs`. For instance, typing `mkfs -t ext3 /dev/hda1` creates an ext3 filesystem on `/dev/hda1`.
5. Mount all of your newly prepared partitions in a subdirectory of the emergency system. For instance, the new root (`/`) partition might be mounted as `/mnt/restore`, the new `/usr` might be mounted as `/mnt/restore/usr`, and so on.
6. Restore *all* the data from your backup. You can do this much like a partial restore, but you'd specify no filenames for restoration. This results in restoring all of the files in the archive. You would also change into the new root partition's mount point on the temporary system (as just described) rather than in the emergency system's root directory.
7. If your partition assignments are different from those of your original installation, edit your restored system's `/etc/fstab` to reflect the changes.
8. Copy your kernel file to a FAT floppy disk. The kernel file is usually `/boot/vmlinuz` (but will be in your restore mount point).
9. Locate a copy of `LOADLIN.EXE`, a DOS program for booting a Linux kernel. Most distributions include this program on their main CD-ROM. Copy this program to the same disk as your kernel file.
10. Prepare a DOS boot floppy. If you don't have a copy of DOS handy, download FreeDOS (<http://www.freedos.org>)

11. Reboot to DOS.
12. Type **FDISK /MBR**. This creates a standard DOS master boot record (MBR), which is necessary for some Linux boot loader configurations and won't do any harm for others. (If you're using a third-party boot loader and you haven't actually replaced your hard disk, though, this step could cause problems and so should be skipped.)
13. Insert the kernel floppy and type **A:\LOADLIN A:\VMLINUZ root=/dev/hda1 ro**. (Change `/dev/hda1` to the Linux identifier for your root partition.)

At this point, your system should boot and run normally. You'll still need to reinstall your Grand Unified Boot Loader (GRUB) or Linux Loader (LILO) boot loader, though. Before proceeding, you may need to edit your GRUB or LILO configuration file. Specifically, if your partition identifiers have changed, you must update them in these files, which are `grub.conf` or `menu.lst` in the `/boot/grub` directory for GRUB and `/etc/lilo.conf` for LILO. Typing **grub-install** should actually install GRUB in the computer's boot sector, while typing **lilo** will do so for LILO.

Depending on your reason for performing a full restore, not all of these steps will be necessary. For instance, you might not need to repartition the disk if the über-gunk that triggered this procedure was purely software-related, such as severe filesystem corruption caused by user error or a buggy program.



- a or -all option, 45
- anewer *file* option, 56
- /boot, 278, 281
- c option, 46, 193
- cnewer *file* option, 56
- /dev, 37
- empty option, 56
- /etc
 - backing up, 32–33
 - files and subdirectories in, list of, 31–32
 - Linux and, 9
 - program configurations in, checking, 33
 - startup scripts and configuration files in, location of, 31
 - X configuration file in, checking, 294
- /etc/crontab, 31
- /etc/cups, 31
- /etc/fstab, 31
- /etc/group, 31
- /etc/inetd.conf, 31
- /etc/inittab, 30, 32
- /etc/modules.conf, 32, 36–37
- /etc/passwd, 32
- /etc/printcap, 32
- /etc/shadow, 32
- /etc/xinetd.conf, 32
- gid *gid* options, 56
- h or -H option, 44
- h or -human-readable option, 46
- /home, 29
- i option, 44, 58
- iname, 56
- k or -keep-going option, 52
- l option, 44, 58
- lname, 56
- m option, 44
- max-depth=*n* option, 45
- maxdepth *levels*, 56
- /media, 29
- mindepth *levels*, 56
- /mnt, 29
- mount option, 56
- name *pattern* option, 56
- perm *mode* option, 56–57

- r , -R, or —recursive option, 58
- s and -summarize options, 45
- S or -separate-dirs option, 46
- size *n* option, 57
- summarize, 45
- t
 - fstype* option, 44
- T option, 44
- /tmp, 29
- type *code* option, 57
- uid *uid* options, 56
- x
 - fstype* option, 44
- z or -uncompress option, 53
- 3D optimizations, enabling in X, 300–302

A

Abnormal termination, 164

Accounts

- creating and deleting, 219–226
- default shells for, identifying, 224
- deleting unused, 24
- GUI tools, managing with, 219–222
- gunk, problems with, 8
- gunk-free, creating, 8
- logging out, importance of, 25
- multiple user, 210
- necessary and unnecessary, 212–218
- network, using, 211–212
- passwords for, setting, 226–233
- removing, 224
- security and, 211
- server, 217–218
- simultaneous user, 210
- special system, 215–217
- suspicious, checking for, 261
- system, creating, 224
- system administration, deleting and restoring, 215
- system administration of, 214–215
- understanding, 210–212
- user, 218

322 Add Group button

- user, tracking, 210–211
- user files in, deleting, 224–226
- Add Group button, 221
- Add User button, 221
- adm** accounts, 215
- Advanced Packaging Tool (APT), 34, 137
- Advertisements, blocking pop-up, 237
- Anti-aliasing, 303
- Appearance & Themes section, configuring in KDE, 80
- Applets, 70
- Application-specific fonts, 303
- at** account, 216
- Attachments, unreadable, 252

B

- Backends, 277
- Backgrounds
 - configuring, 71
 - setting, 76–77
- Backups
 - for **/etc**, 32–33
 - types of, 25–26
- .bak**, 50
- Balsa, 126
- Bandwidth thieves, blocking, 237
- .bash_history**, 64
- .bash_logout**, 64
- .bash_profile**, 64
- .bashrc**, 64
- Berkeley Standard Distribution Line Printer Daemon (BSD LPD), 169
- bin** account, 216
- Bit depths, high and low, 296
- Blackbox, 90
- Bogofilter, 245
- Boot loaders, 30, 184–185, 279
- Boot messages, 186
- Boot process, 184–193
- Boot sector, 184–185
- Boot time, 3
- Broadband firewalls, 257–258
- Broadband routers, 257–258
- Bugs
 - desktop environments and, 167
 - program, working around, 166–167
 - reporting, 167–168
 - software, fixing, 147

C

- .c**, 50
- .c++**, 50
- .cc**, 50
- Central processing units (CPUs). *See also* Computers
 - bad, 267

- load hogs on, locating, 194–196
- memory hogs on, locating, 197
- system requirements, 161
- Channels, 148–149
- Child, 186
- chkconfig**, 189
- chkrootkit**, 261
- Clutter, eliminating, 29
- Code bloat, 157–158
- Color depths
 - common settings for, list of, 295
 - setting, 295–297
 - X, changing in, 296–297
 - X, setting in, 295–297
- Common Unix Printing System (CUPS)
 - browsing feature of, 178–179
 - configuration tools, 171–173
 - understanding, 169–171
- Computers. *See also* Central processing units (CPUs)
 - eye candy, effects from, 159
 - gunk-free versus gunked-up, 3–4
 - monitoring, 24–25
 - performance problems with, 156–164
 - securing (physically), 24
 - system requirements, 161–162
- .conf**, 50
- Configurations
 - new, resisting temptation of, 20–23
 - performance problems from, 158–160
 - simplifying, 19–20
- CONT**, 200
- Content, blocking inappropriate, 237
- Control Center, 28, 80–81, 159, 229, 308–309
- Cookies
 - blocking, 237
 - Mozilla Firefox, managing in, 116–117
- Core memory, 51
- .cpp**, 50
- Crackers
 - gunk caused by, 14–15
 - hackers versus, 9
- Crashes. *See also* Hangs
 - causes of, determining, 164–166
 - Linux, 165
 - system, correcting, 266–267
 - triggers for, determining, 166
 - working around, 166–167
- cron** account, 216

D

- daemon** account, 216
- DansGuardian, 239
- Data, backing up, 25–26
- .DCOPServer_***, 64
- .deb**, 50

Debian
 package management system, features of, 135–137
 packages, 34
 startup scripts, location of, 192
 understanding, 11

Degunking Windows, 2

Delete button, 222

Dependencies
 bypassing, 153–154
 defining, 130
 problems with, handling, 152–154
 software, conflicting in, 147
 tracking, 33, 152–153

Desktop environments. *See also* GNOME; K Desktop Environment (KDE)
 bugs and, 167
 components of, 70
 creating own, 92–93
 Linux to recognize, configuring, 93–96
 understanding, 292
 XFce, 86–89

Desktops
 configuring in KDE, 80
 gunk-free, 6–8
 gunked-up, 6–8

Development kernels, 273

Development packages, 144

Device modes, modifying, 267–268

df, tracking disk usage with, 42–44

Dictionary attacks, 227

Directory structures
 creating, 29
 files in, deleting, 29
 gunk and, 65–66

Disk space. *See also* Memory; Random access memory (RAM)
 consumption of, reducing, 131
 system requirements, 161

Disk usage. *See also* Random access memory (RAM)
 tracking, 42–48

Displays, problems with high resolution, 296

Distributions. *See also* under individual distributions
 kernel packages for, upgrading, 272
 list of, 11–13

.dmrc, 64

DocBook, 110

Dot files
 cleaning up, 62–68
 common, list of, 64–65
 deleting, 28–29, 66–68
 editing, 6
 gunk and, 65–66
 as gunking agents, 6
 understanding, 6
 viewing, 63

dpkg, 34, 135–137

DRI, 301

Drivers
 configurations for, changing, 37
 degunking, 36–37
 gunk, 10
 switching, 37
 upgrading, 37

du, tracking disk usage with, 45–46

du -s ./*, 46

Dumpster diving, for passwords, 232

E

E-mail
 cleaning up, 244–252
 degunking tool for, selecting, 244–246
 odd, reading, 250–252
 security and, 123–125

E-mail clients, forms of clutter, 121

Emacs, 126

.emacs.d, 64

Emblems, setting in GNOME, 76–77

Encryption, using, 258–259

Enlightenment, 90–91

.esd_auth, 64

Evolution
 alternatives to, 126
 blocking e-mail gunk with, 246–248
 configuring, 121–123
 degunking, 120–126
 e-mail in, organizing, 123
 fonts in, 122, 252
 headers in, 122
 message preview pane in, 122
 message summary information in, 122
 toolbar in, 122

Eye candy, system performance and, 159

F

Features, problems with unused, 19–20

Fedora
 availability of, 11
 GNOME defaults and, 70
 gunk-free and gunked-up desktops on, examples of, 6–7
 local startup script for, 192
 OpenOffice.org programs in, reconfiguring, 104
 Red Hat Package Management program and, 138–139
 RPM and, 34
 Security Level Configuration tool of, 256
 startup scripts for, location of, 192
 User Manager tool of, 219–220

324 .file, identifying file types

file, identifying file types with, 52–53

File browsers, 74

File browsing, tweaking in KDE, 83–84

File Management Preferences dialog box, setting defaults in, 74–76

File Transfer Protocol (FTP), 258

File types

file, identifying with, 52–53

strings, examining with, 53–54

Filename extensions, list of, 50–51

Files

deleting, 6, 61–62

extensions, identifying by, 50–52

monitoring changes to, 260–261

OpenOffice.org, compatibility in, 108–109

organizing, 6, 59–61

searching for, 55–59

sharing, 259

transferring, 258

Files types, identifying, 54–55

FilterProxy, 239

find, using, 55–57

Firefox Fonts & Colors dialog box, 111–113

Firefox Preferences dialog box, 111–112

Firestarter, 257

Firewalls, using, 255–257

Font directories

X, adding, 307–308

X, preparing, 305–307

Font smoothing, 303

Fonts

commercial collections of, 304

in commercial software, 304

degunking, 97

in Evolution, 122

foreign, reading, 252

GNOME, configuring, 71–72

Linux, understanding, 303

low-cost or free, 304

managing, 310–311

Microsoft core, 304

Mozilla Firefox, improving in, 111–113

obtaining, 303–305

OpenOffice.org, managing in, 104–106

performance with, degrading, 159

program, degunking, 26

ugly or poorly sized, reading, 252

X, configuring in, 302–311

Xft, configuring in, 308–310

.fonts, 64

.fonts.cache-1, 64

.fonts.conf, 64

Foomatic, 277

Framebuffer, 36, 294

FVWM, 91

FVWM '95, 91

C

Galeon, 119

games account, 216

.gconf, 64

.gconfd, 64

Gecos field, 213

Gentoo

startup scripts, location of, 192

understanding, 11–12

Ghostscript, 36, 276–277

GID field, 213

.gif, 50

GIMP, Preferences dialog box from, 27–28

.gimp-1.2, 64

.gimp-2.0, 64

GIMP-Print, 277

GNOME

applications, configurations for, 64

backgrounds and emblems in, setting, 76–77

configuration for, starting with new, 78–79

configuration tools for, 27–28

Control Center, 28

desktop environments to, alternative, 85–96

desktops uncluttered in, keeping, 78

feature performance of, tweaking, 70–74

Fedora and defaults for, 70

file browsing in, tweaking, 74–77

file browsing options and, 76

office suite for, 109

popularity of, 21, 70

Preferences window in, 70–72

RAM, consumption of, 86

.gnome, 64

GNOME Disk Usage (GDU), 46–47

GNOME Office, 109

GNOME RPM, 138

.gnome2, 64

.gnome2_private, 64

GNU Network Object Model Environment (GNOME). See GNOME

Grand Unified Boot Loader (GRUB), 184–185, 279–281

Graphical user interface (GUI)

accounts with tools, managing, 219–222

cross-platform tools, 114

disk usage with tools, monitoring, 46–48

dragging-and-dropping in environments, 60

firewalls, 257

improving, 97–98

kernel with tool, configuring, 273–274

login tools, 93–94

Macintosh and, 157

package management tools, 138

passwords with tools, setting, 229–230

processes with tools, killing, 199–201

subsystem, components of, 292–293

grep, using, 57–59

Group ID number, modifying processes by, 206

Groups

default, 223

user-specific, creating, 224

users to, adding, 223

users to, assigning, 223

.gtkrc, 64

GuardDog, 257

Gunk

adding by mistake, 14

adding maliciously, 14–15

defining, 3

directories and, 65–66

dot files and, 65–66

eliminating, 15–16

hardware or drivers, 10

Linux to, susceptibility of, 2

out-of-the-box, 11–13

recognizing and determining, 2–5

system configuration, 8–10

user configuration, 5–8

GunkBuster's Notebook

"Adding a Program Launcher with XFce," 88

"Configuring Multiple Applications," 27–28

"Dealing with Dependencies," 145–146

"Degunking with the Linux **root** Account," 9

"NAT Appliances," 257–258

"One Person's Gunk Is Another Person's Treasure," 4–5

"Perils of Degunking the Web," 243–244

"Recompiling Software to Improve Performance," 162–164

"Setting Backgrounds and Emblems," 76–77

"Setting HTML and Security Options," 123–125

"Setting the Resolution for LCDs," 298

"Using Network Accounts," 211–212

"Using Precompiled Kernel Modules," 282

"When Complexity Helps," 22–23

"Working with File Types," 51–52

.gz, 50

gzip compression system, 53

I

.h, 50

Hackers, crackers versus, 9

halt accounts, 215

Hangs, 164, 269. *See also* Crashes

Hard disks, speeding up, 268–269

Hardware

broken, 35

crashes caused by, 165

degunking, 34–36

flaky, 35

gunk, 10

inadequate, 34, 161–162

incompatible, 35

informational web sites on, 289

Linux-compatible, identifying, 287–289

non-kernel drivers and, 276–277

problems with, common, 266–270

procedures for, troubleshooting, 264–266

replacing, 286–289

swapping out, 266

troubleshooting, 264–270

Hardware drivers

installing, 277–286

package upgrade for, locating, 272

sources for, locating, 271–277

Header packages, 144

Headers, in Evolution, 122

High-level languages, 156

Home directory

users, changing for, 223

users, creating for, 223

Home directory field, 213

Horizontal refresh rate, 299

.htm, 50

.html, 50

HUP, 200

Hypertext Markup Language (HTML)

bad, 236

e-mail, reading, 251

Hypertext Transfer Protocol (HTTP), 258

I

IceWM, 89–91

info info, 38

Info pages, 38

init, 30, 186–187

INT, 200

Internet & Network section, configuring in KDE, 81

Internet Mail Access Protocol (IMAP), retrieving mail with, 123

Intrusions, monitoring for, 259–261

J

Java

buggy or malicious, 237

security and, 114–116

JavaScript

buggy or malicious, 237

security and, 114–116

.jpg, 50

K

K Desktop Environment (KDE)

Appearance & Themes section of, configuring, 80

- Components section of, configuring, 81
- configuration for, starting with new, 85
- configuration tools for, 27–28
- Control Center, 28, 80–81, 159, 229, 308–309
- desktop environments to, alternative, 85–96
- desktops uncluttered in, keeping, 84–85
- feature performance of, tweaking, 79–82
- file browsing in, tweaking, 83–84
- Internet & Network section of, configuring, 81
- KDiskFree and, 46
- KMail and, 126
- KOffice and, 109
- Peripherals section of, configuring, 82
- popularity of, 21, 70
- Power Control section of, configuring, 82
- RAM, consumption of, 86
- Regional & Accessibility section of, configuring, 82
- Security & Privacy section of, configuring, 82
- Sound & Multimedia section of, configuring, 82
- System Administration section of, configuring, 82

.kde, 64

- KDirStat, 47–48

- KDiskFree, 46–47

- Keep it Simple, Stupid (KISS) principle, 20

- Kernel drivers, installing, 277–282

- Kernel modules

- installing, 279

- precompiled, using, 282

- Kernel ring buffer, 265

- Kernels

- booting new, 279–281

- configuring, 272–275

- drivers, sources for, 271–272

- modules, third-party, 275–276

- operating system, understanding, 2

- overwriting files, 278

- recompiling, 13, 272, 278

- understanding, 271–275

- Keyboards, configuring GNOME, 72

- KILL, 200

- kill, 201

- killall, 201–202

- KMail, 126

- Kneifilter, 257

- KOffice, 109

- Konqueror, 119–120

- Kooka, 285

- ksysv, 190

L

- LaTeX, 110

- less, 55

- Libraries, crashes and, 165

- Library packages, 144

- Line printers, 170

- Linspire, using root accounts in, 10

- Linux

- 12-step degunking program for, 39

- complexity of, minimizing, 131

- degunking strategy for, 18–19

- desktop environments for, 70

- directory system, 29

- distributions, 2, 11–13

- filename extensions, list of, 50–51

- filesystem, making changes in, 37

- fonts in, understanding, 303

- GUI in, improving, 97–98

- gunk, susceptibility to, 2

- information sources for, 37–39

- operating system kernel, understanding as an, 2

- package management systems for, 33–34, 133–137

- password files for, 212–214

- printing in, understanding, 168–171

- security of, ensuring, 24–25

- signal numbers and names, list of, 200

- software for, flexibility of, 4

- startup procedure for, intervening in, 188–193

- startup procedure for, understanding, 184–193

- system operations, understanding, 30–33

- system requirements for, 161–162

- viruses and, 2, 123–124

- web browsers, list of, 119–120

- Web server software under, running, 4

- worms and, 2, 123–124

- Linux Loader (LILO), 184–185, 279–281

- Linux user groups (LUGs), 39

- Load averages, 194

- Local Address** column, 254

- Local startup scripts, 192–193

- Log files, monitoring, 260, 265

- Logins, remote, 258

- lp account, 216

- LPD printing systems, 169–170, 174–175

- LPRng system, 169

- Lynx, 120

- LyX, 110

M

- .macromedia**, 64

- Magazines, for Linux, 38–39

- Mail** account, 216

- .Mail** or **.mail**, 64

- Mail servers

- packages, spam-fighting tools and, 245–246

- Simple Mail Transfer Protocol (SMTP) and, 259

- Mailfilter, 245–246

- Mailing lists, 38

- man** account, 216

man man, 37
man pages, 37
 Mandrake
 startup scripts, location of, 192
 understanding, 12
 xfs font server and, 144
 Memory. *See also* Disk space; Random access memory (RAM)
 locating CPU hogs, 197
 Menus, increasing or decreasing options in, 27
.metacity, 64
 Mice
 GNOME, configuring, 72
 system requirements, 162
mkdir, 59
 Monitors, system requirements for, 162
.mozilla, 64
 Mozilla Firefox
 alternatives to, 119–120
 cookie management tools in, 116–117
 degunking, 110–120
 Firefox Preferences dialog box, 111–112
 Fonts & Colors dialog box, 111–113
 fonts in, improving, 111–113
 password management in, 117–118
 security in, improving, 113–119
 SSL encryption in, 118–119
 Mozilla Firefox print dialog box, 168–169
 .mp3, 50
 Multimedia, system performance and, 159
 Mutt, 126
 MWM, 91

N
 Nautilus
 File Management Preferences dialog box of, 74–76
 working with, 74–76
.nautilus, 64
 Netscape, 119
netstat, 253–254
network, 189
 Network address translation (NAT) appliances, 257–258
 Network connections, repairing, 269–270
 Network installations
 launching, 102–103
 performing, 101–102
 Network security, improving, 252–261
 Network sniffing, for passwords, 231
networking, 189
 Networks, problems with gunk from, 8–9
news account, 216
 Newsgroups, 38
nice, 204–205
nobody account, 216
ntsysv, 189
 nVidia, 301

O
 .o, 50
 .ogg, 50
 OpenOffice.org
 alternatives to, 109–110
 degunking, 100–110
 file compatibility in, 108–109
 fonts in, managing, 104–106
 installation options for, 100–104
 printers in, managing, 107–108
 Select Installation Type dialog box, 103–104
 Opera, 120
operator accounts, 215
 Overheating, problems with hardware, 267

P
 Package browsers, options for, 138
 Package descriptions
 additional information on, locating, 144–145
 interpreting, 143–144
 Packages, 30
 cleaning out unused, importance of, 130–133
 defining, 130
 dependent, deleting, 152
 dependent, upgrading or installing, 152
 disk space consumption of, reducing, 131
 management systems for, understanding, 33–34, 133–137
 management tools for, using, 133–143
 removing unused, 131
 requiring, reasons for, 145–146
 security risks from, minimizing, 132
 testing, 145–146
 unused, identifying, 143–146
 upgrade complications from, minimizing, 132–133
 Pagers, degunking desktops with, 97
 Parent, 186
passwd, using, 230
 Password field, 213
 Passwords
 account, setting, 226–233
 cracking, 231
 creating, cautions when, 228–229
 creating, tips for, 227–228
 discovery methods for, 227, 230–233
 dumpster diving for, 232
 files, perusing, 212–214
 GNOME, changing, 72
 GUI tools, setting with, 229–230
 identical, using, 231
 length of, 227
 Mozilla Firefox, managing in, 117–118
 network sniffing for, 231
 protecting, 230–233
 secure, selecting, 226–229

328 pattern

- shoulder surfing for, 231
- social engineering, 227, 232
- storing, 232
- strong, importance of, 25
- user, protecting, 226

pattern, 56

- .pdf, 50

Performance. *See also* Random access memory (RAM)

- computer, problems with, 156–164
- software to improve, recompiling, 162–164
- system requirements, 161–162

Peripherals section, configuring in KDE, 82

Phishing, 231–232

PID/Program name column, 254

- .pl, 50
- .png, 50

Ports, blocking, 255–257

Post Office Protocol (POP), retrieving mail with, 123

postmaster accounts, 215

PostScript printers, 170

Power Control section, configuring in KDE, 82

Power supply, problems with, 267

Printer drivers

- installing, 284
- sources for, 276–277

Printers

- categories of (per Linux Printing Web site), 179–180
- compatible, selecting, 175
- default resolution for, 177
- defining, 172–173
- degunking, 171–179
- drivers for, downloading, 175
- drivers for, selecting, 175–176
- LPD options for, selecting, 174–175
- OpenOffice.org, managing in, 107–108
- options for, setting, 176–177
- output devices for, selecting, 173–175
- ports for, determining, 173
- resolution for, setting, 176–177
- selecting, 179–181
- USB printer options for, selecting, 174
- Windows options for, selecting, 174–175

Printing

- in Linux, 168–171
- network, enabling, 178–179
- performance, improving, 168–181

priority, 204

Privoxy

- running, 239–240
- understanding, 239
- using, 240–243

Process ID number, modifying processes by, 206

Process priority

- adjusting, 202–206
- defining, 202–203
- identifying, 203–204
- non-standard, starting with, 204–205

- running, modifying, 205–206

Processes

- name, finding by, 198–199
- stray, killing, 199–202
- suspicious, checking for, 261

Procmail, 245

Program configurations

- broken, 27
- checking, 33
- degunking, 26–29
- simplifying, 27

Program languages, problems with, 156–157

Programs

- abnormal termination of, 164
- bugs in, working around, 166–167
- configurations tools of, using, 27
- configuring effort for, 20
- daily use problems with, 20
- eliminating unnecessary, 24
- features in, complexity of, 157
- hanging, 164, 269
- interference from, problems with, 160–161
- learning curve of, 20
- locating misbehaving, 193–199
- memory consumption of, 157
- new, resisting temptation of, 20–23
- orphaned, 186
- poor options for, 158
- running too many, problem of, 158
- simplicity and complexity in, balancing, 22–23
- stability of, improving, 164–168
- unused, problems with, 19–20
- updating, 24

Properties button, 221–222

Proto column, 254

ps, 198, 203

- .ps, 51
- .py, 51

Q

QUIT, 200

R

Random access memory (RAM). *See also* Core memory; Memory

- adding, 35
- available, increasing factor of, 156
- bad, 35, 165, 264, 267
- bit depths and, 296
- GNOME consumption of, 86
- hardware and, 34
- KDE consumption of, 86
- overheating, 267
- rationing, 160
- replacing, 36

- swap space and, 197
- system requirements for, 161
- video cards and, 296
- wasted resources and, 19
- XFce consumption of, 86
- rc-update**, 189
- Reboots, 98, 165, 200, 266, 268, 297
- Red Hat
 - local startup script for, 192
 - package management system for, using, 138–141
 - RPM and, 34
 - Security Level Configuration tool of, 256
 - startup scripts, location of, 192
 - understanding, 12
 - Update Agent, 65
 - User Manager tool of, 219–220
 - xfs** font server and, 144
- Refresh rate, setting in X, 299–300
- Regional & Accessibility section, configuring in KDE, 82
- Removable storage, changing in GNOME, 72
- renice**, 206
- Resolution
 - default printer, 177
 - LCDs, setting for, 298
 - printer, setting, 176–177
 - X, setting in, 297–298
- .rhn-applet**, 65
- .rhn-applet.conf**, 65
- rm**, 62
- root**
 - Linux and, 9
- Root Accounts
 - degunking with, 9
 - Linspire, using in, 10
- Root kits, 15, 261
- rpm**, 34, 134–135
- .rpm**, 51
- RPM Package Manager (RPM)
 - creator of, 34
 - Fedora and, 34
 - package management system, features of, 134–135
- Runlevels
 - default, degunking, 191
 - defining, 30, 187
 - services from, enabling or disabling, 190
- S**
- Scanner Access Now Easy (SANE), 36, 277
- Scanner drivers
 - installing, 284–286
 - sources for, 277
- Screen resolution, configuring GNOME, 73
- Screensavers, configuring GNOME, 73
- Script kiddies, 15
- Secondary boot loader, 184–185
- Security. *See also* Passwords; Viruses; Worms
 - E-mail and, 123–125
 - Java and, 114–116
 - JavaScript and, 114–116
 - Linux, 24–25
 - patches, updating with, 24
 - programs, threats from, 20
 - software updates and, 147
 - wireless networks and, 259
- Security & Privacy section, configuring in KDE, 82
- Select Installation Type dialog box, 103–104
- Server Message Block/Common Internet File System (SMB/CIFS), 259
- Server programs, 144
- Servers
 - counting, 253–254
 - identifying, 254–255
 - intrusions on, checking for, 261
 - removing unnecessary, 252–255
 - shutting down, 24
 - tracking down, 255
- Service Configuration tool, 190
- .sh**, 51
- Shell field, 213
- Shoulder surfing, for passwords, 231
- shutdown** accounts, 215
- Siag Office, 109
- Signal names, list of, 200
- Signal numbers, list of, 200
- .signature**, 65
- Simple Mail Transfer Protocol (SMTP), 259
- Skeleton files, copying, 223
- Slackware
 - startup scripts, location of, 192
 - SysV startup scripts and, 187
 - understanding, 12
- .so**, 51
- Software
 - bug fixes for, 147
 - dependencies in, conflicting, 147
 - flexibility of, 4
 - improvements to, 147
 - new, updating with, 147
 - performance problems with, 156–158
 - problems with, 8
 - security updates for, 147
 - updating, 146–151
- Sounds, configuring GNOME, 73
- Source code, problems with, 157
- Spam, reporting, 250
- Spam blockers, 244–250
- SpamAssassin
 - configuration of, tweaking, 248–250
 - spam filtering with, 246–248
- Squid, 238
- SquidGuard, 238
- .ssh**, 65

330 SSL encryption, in Mozilla Firefox

SSL encryption, in Mozilla Firefox, 118–119
Stable kernels, 273

Startup procedure

intervening in, 188–193

understanding, 184–193

Startup scripts

defining, 187

locations of, 192

SysV, configuring for, 188–191

Sticky bit, 226

STOP, 200

strings, examining files with, 53–54

Styles, simplifying, 97

Super server, 187

SuSE

startup scripts, location of, 192

understanding, 12

Swap space

RAM and, 197

.sxc, 51

.sxw, 51

Sylpheed, 126

Synaptic

software updates, using for, 150–151

using, 141–143

sync accounts, 215

System Administration section, configuring in
KDE, 82

System backups, 25–26

System configuration, gunk, 8–10

System operations, understanding, 30–33

System resources, unused, 19–20

System software, degunking strategies for, 30–34

System V, 187

SysV, configuring startup scripts for, 188–191

sysv-rc-conf, 189

SysV startup scripts

creating or modifying, 192

understanding, 30

T

.tar, 51

.tbz, 51

.tcl, 51

TERM, 200

TeX, 110

.tgz, 51

The Linux Documentation Project, 38

The Linux Hardware Compatibility HOWTO
document, 288

.themes, 65

Themes, configuring GNOME, 73

Thrashing, 197

Thunderbird, 126

top, 195–201

Triggers, determining, 166, 265

TWM, 91

.txt, 51

U

UID field, 213

UIDs, synchronizing, 224

Update Agent, using for software updates,
148–150

uptime, 194–195

USB printers, selecting options for, 174

Usenet, 38

User configuration gunk, 5–8

User data backups, 25

User files, degunking, 26–29

User groups, for Linux, 39

User interfaces, complexity of, 157

User Manager program

button bar buttons in, 220–222

of Red Hat and Fedora, 219

useradd, using, 222–224

userdel, using, 224

Username, modifying processes by, 206

Username field, 213

USR1, 200

USR2, 200

Utah GLX, 301

uucp account, 216

V

Vertical refresh rate, 299

Video cards, system requirements for, 161

Viruses, 2, 123–124

W

.wav, 51

Web

degunking the, 236–244

transactions on, 258

Web forums, for Linux, 38

Web proxy

degunking, using, 238–244

selecting, 238–239

Web server software, running under, 4

Web sites

filterproxy.sourceforge.net, 239

<http://apt4rpm.sourceforge.net>, 138, 146

<http://balsa.gnome.org>, 126

<http://blackboxwm.sf.net>, 90

<http://bogofilter.sourceforge.net>, 245

<http://config.privoxy.org/show-status>, 243

<http://corefonts.sourceforge.net>, 304

<http://enlightenment.org/pages/main.html>, 91

<http://expansa.sns.it/knetfilter>, 257

<http://fedora.redhat.com>, 11

<http://folding.stanford.edu>, 192

<http://galeon.sourceforge.net>, 119

<http://gfontview.sourceforge.net>, 305

<http://gimpprint.sourceforge.net>, 277

<http://groups.google.com>, 265

- <http://hpwww.ec-lyon.fr/vincent/>, 46–47
- <http://kdirstat.sourceforge.net>, 47
- <http://kmail.kde.org>, 126
- <http://knode.sourceforge.net>, 38
- http://linux.tucows.com/email_clients_default.html, 126
- <http://lynx.browser.org>, 120
- <http://mailfilter.sourceforge.net>, 246
- <http://pan.rebelbase.com>, 38
- <http://siag.nu>, 109
- <http://spamassassin.apache.org>, 246
- http://spamassassin.apache.org/tests_3_0_x.html, 249
- <http://sylpheed.goodday.net>, 126
- <http://themes.freshmeat.net>, 73
- <http://tldp.org>, 38
- <http://utahglx.sourceforge.net>, 301
- <http://www.sun.com/software/star/staroffice/>, 100
- <http://www.sun.com/software/learnabout/java/>, 114
- <http://xdiskusage.sourceforge.net>, 47
- <http://xwinman.org>, 92
- www.1001freefonts.com, 304
- www.adobe.com, 304
- www.all-daybreakfast.com/wm2/, 91
- www.bitstream.com/fonts, 304
- www.chkrootkit.org, 214, 261
- www.corel.com, 304
- www.cs.wisc.edu/ghost/, 36, 170, 276
- www.ctan.org/tex-archive/fonts/, 304
- www.cups.org, 169
- www.dansguardian.org, 239
- www.debian.org, 11
- www.debian.org/doc/manuals/apthowto/, 137
- www.exploits.org/v4l/, 289
- www.fontfreak.com, 304
- www.free-fonts.com, 304
- www.freedos.org, 265
- www.fs-security.com, 257
- www.fvwm.org, 91
- www.gentoo.org, 11, 163
- www.gnome.org, 70
- www.gnome.org/gnome-office/, 109
- www.gnu.org/software/emacs/emacs.html, 126
- www.gnupg.org, 259
- www.icewm.org, 91
- www.kde.org, 70
- www.kde.org/apps/kooka/, 285
- www.kernel.org, 272, 275
- www.koffice.org, 109
- www.linmodems.org, 289
- www.linotype.com, 304
- www.linux-mag.com, 38–39
- www.linux.com, 38
- www.linuxjournal.com, 38
- www.linuxmafia.com/wpfaq/downloadwp8.html, 110
- www.linux.org, 38
- www.linux.org/apps/, 93, 158
- www.linux.org/groups/, 39
- www.linuxprinting.org, 175, 179, 289
- www.linuxprinting.org/foomatic.html, 277
- www.lyx.org, 110
- www.mandrakelinux.com, 12
- www.memtest86.com, 165
- www.mozilla.org, 110
- www.mozilla.org/products/thunderbird/, 126
- www.mutt.org, 126
- www.myfonts.com/WhatTheFont/, 305
- www.netscape.com, 110
- www.novell.com, 12
- www.novell.com/products/desktop/features/evolution.html, 120
- www.nvidia.com, 277, 301
- www.openoffice.org, 100
- www.opera.com, 120
- www.privoxy.org, 238–239
- www.redhat.com, 12
- www.rodsbooks.com, 112
- www.rpmfind.net, 141, 153
- www.samag.com, 39
- www.sane-project.org, 36, 277, 289
- www.simonzone.com/software/guarddog/, 257
- www.slackware.com, 12
- www.squid-cache.org, 238
- www.squidguard.org, 238
- www.suse.com, 12
- www.tldp.org/HOWTO/Hardware-HOWTO/, 288
- www.tripwire.com, 25
- www.tripwire.org, 260
- www.webmin.org, 219
- www.windowmaker.org, 91
- www.winehq.org, 120
- www.xandros.com, 13
- www.xfce.org, 86
- www.xfree86.org, 276, 293
- www.xig.com, 294
- www.x.org, 276, 289, 293
- www.xsane.org, 285
- Whitelists, 249–250
- Widget sets, 292
- Window Maker, 91
- Window managers
 - bare, using, 89–92
 - defining, 70
 - understanding, 144, 292
- Windows printers, selecting options for, 174–175
- .wine**, 65
- Wireless networks, security and, 259
- WM2, 91
- WordPerfect, 110
- Worms
 - blocking, 244–250

Linux and, 2, 123–124
 as network gunk, 8
 Windows, 251–252

X

3D optimizations in, enabling, 300–302
 color depth in, setting, 295–297
 color depth of, changing, 296–297
 drivers for, selecting, 294
 font directories in, preparing, 305–307
 font directories to, adding, 307–308
 fonts in, configuring, 302–311
 performance and appearance of, improving, 295–302
 refresh rate in, setting, 299–300
 resolution in, setting, 297–298
 role of, 292–294
 working with, 10

X core fonts
 configuring, 305–308
 understanding, 303

X drivers
 framebuffer drivers and, 36
 installing, 282–284

x or **-one-file-system** option, 46

X server drivers, sources for, 276

X servers, system performance and, 158–159

X Window System. *See* X

Xandros, 13

.Xauthority, 65

.Xclients, 65

.Xclients-default, 65

.Xdefaults, 65

xdev option, 56

XDiskUsage, 47, 49

XFce
 Program Launcher with, adding a, 88
 RAM consumption of, 86
 using, 86–89

.xfce4, 65

XFree86, 276, 293

Xft fonts
 configuring, 308–310
 understanding, 303

.xinitrc, 65

.xms, 65

X.org-X11, 276

XSane, 285

Y

YaST, 190, 257
 YaST2, 190, 257

Z

.zip, 51

Zombie processes, 164–165